ARI Contractor Report 96-04

An Intelligent Flight Trainer for Initial Entry Rotary Wing Training

Sandeep S. Mulgund, Greg L. Zacharias, and Mehran Asdigha

Charles River Analytics

May 1996

United States Army Research Institute for the Behavioral and Social Sciences

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction
of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Director

NOTICES

DISTRIBUTION: This report has been cleared for release to the Defense Technical Information Center (DTIC) to comply with regulatory requirements. It has been given no primary distribution other than to DTIC and will be available only through DTIC or the National Technical Information Service (NTIS).

FINAL DISPOSITION: This report may be destroyed when it is no longer needed. Please do not return it to the U.S. Army Research Institute for the Behavioral and Social Sciences.

NOTE: The views, opinions and findings in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other authorized documents.

DRAFT SF 298

1. Report Date (dd-mm-yy)	2. Report Type	3. Dates covered (from... to)
May 1996		FINAL July 1993-December 1995

4. Title & subtitle	5a. Contract or Grant #
An Intelligent Flight Trainer for Initial Entry Rotary Wing Training	MDA903-93-C-0132
	5b. Program Element # 0605502
6. Author(s)	5c. Project # 770
Sandeep S. Mulgund, Greg L. Zacharias, and Mehran Asdigha	5d. Task # 6901
	5e. Work Unit # C06

7. Performing Organization Name & Address	8. Performing Organization Report #
Charles River Analytics 55 Wheeler Street Cambridge, MA 02138	R92351
9. Sponsoring/Monitoring Agency Name & Address U.S. Army Research Institute for the Behavioral & Social Sciences ATTN: PERI-IR 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. Monitor Acronym
	11. Monitor Report # Contractor Report 96-04

12. Distribution/Availability Statement

Approved for public release; distribution is unlimited.

13. Supplementary Notes This report is published to meet legal and contractual requirements and may not meet ARI's scientific or professional standards for publication. COR: John A. Dohme

14. Abstract

This report summarizes a Phase II SBIR effort to develop and validate an Intelligent Flight Trainer (IFT) for use in Initial Entry Rotary Wing Training at Fort Rucker, AL. The IFT is a simulator-based system designed to assist students in developing proficiency on a suite of initial entry rotary wing maneuvers. The IFT encapsulates instructor pilot domain knowledge in an expert system shell that provides tutorial and performance monitoring functions through a synthetic voice generator. The expert system shell works in concert with a variable stability augmentation control law that makes it easier for a neophyte student to control the motion of the simulated vehicle. A four-task development, demonstration, and validation process was accomplished.

15. Subject Terms

Intelligent training systems
Simulation
Helicopters

Security Classification of			19. Limitation of Abstract	20. # of Pages	21. Responsible Person (Name and Telephone #)
16. Report	17. Abstract	18. This Page			
Unclassified	Unclassified	Unclassified	Unlimited		Charles A. Gainer, (334)255-2834

Charles River Analytics

Final Report No. R92351
Contract No. C9235

An Intelligent Flight Trainer for Initial Entry Rotary Wing Training

Sandeep S. Mulgund, Mehran Asdigha, and Greg L. Zacharias

Charles River Analytics
55 Wheeler Street
Cambridge, MA 02138

December 12, 1995

Prepared for:

Jack A. Dohme
ARI Rotary Wing Aviation Research Unit
PERI-IR
Fort Rucker, AL 36362-5354

PREFACE

This report provides technical information on the development of the intelligent flight trainer (IFT). The report may be viewed within the context of a series of low-cost training simulator developments undertaken by the Army Research Institute Rotary Wing Aviation Research Unit (RWARU) at Fort Rucker, Alabama. The training concept being evaluated is to provide Army student pilots with initial entry helicopter flight training in a low-cost automated simulator. The end goal of this research and development program is to reduce the cost of initial entry rotary wing (IERW) training while simultaneously improving training safety and training efficiency.

Prior research and development conducted by the RWARU demonstrated that a low-cost visual simulator could provide significant positive transfer-of-training (TOT) using Army flight students as research subjects. Additional research demonstrated that training in a low-cost simulator could be substituted for training in the aircraft and that basic hovering skills could be trained in the simulator using an expert system in place of a dedicated instructor pilot (IP). The latter development, termed the automated hover trainer (AHT), demonstrated significant positive TOT to the UH-1 aircraft for the five hovering maneuvers trained in the simulator.

The pioneering engineering efforts that led to the AHT and IFT were accomplished by the University of Alabama Flight Dynamics Laboratory and Microcomputer Laboratory. The effort detailed in this report reconfigured the AHT to run on personal computer architecture and went beyond automated hover training to include automated training in traffic pattern flight. To date, the engineering developments reported upon in the present document have not been subjected to a research evaluation. There is a need to perform an experimental evaluation of the IFT in order to validate its training value and conduct a TOT from the trainer to the aircraft using Army aviator candidates.

ACKNOWLEDGMENT

This work was performed under Contract MDA903-93-C-0132 with the U.S. Army Research Institute at Fort Rucker, AL. The authors thank the technical monitor, Dr. Jack A. Dohme, for his support and direction on this project. We also thank Messrs. Ken Persin, Larry Murdock, Mike Couch, and Dale Weiler at Fort Rucker for their tremendous technical contributions to the development of the Intelligent Flight Trainer. Finally, we owe a debt of gratitude to Professors Kalmanje Krishnakumar and Jim Dudgeon at the University of Alabama for their efforts in developing the UH-1 simulation model and the image generator interface software.

EXECUTIVE SUMMARY

The Army Research Institute (ARI) has developed and demonstrated the UH-1 Training Research Simulator (TRS), and shown it to be effective in providing low-cost, effective simulator-based basic training of initial-entry rotary wing (IERW) maneuvers. The system integrates technologies of distributed computational processing and computer image generators in a configuration that has the potential for minimizing dependence on a dedicated instructor pilot. Six empirical studies have demonstrated the system's effectiveness in delivering positive *transfer-of-training* to the UH-1 helicopter, and further studies are planned to expand the evaluation effort's scope.

A need has existed, however, to improve the instructional portions of the system by incorporating instructor pilot (IP) domain knowledge, and by providing the student pilot (SP) with appropriate tutorial feedback regarding proficiency progress. By doing so, significant savings in IP hours could be realized to complement the savings already achieved through the low-cost hardware configuration. Of perhaps greater significance was the potential of demonstrating how intelligent tutoring systems (ITSs) could be hybridized with flight training simulators to support the development of a new generation of low-cost *intelligent flight trainers* (IFTs). The potential thus existed far beyond the UH-1 TRS, and may have significant impact on several other existing and planned flight trainers.

In order to accomplish these objectives, we proposed that the IP's domain knowledge be encapsulated in the form of an expert system (ES), and that the entire system be hosted a PC-based distributed environment. Feedback would be provided to the student verbally via a synthetic voice generator that was already a component of the existing system. Furthermore, the presence of an expert system shell within the system could support the implementation of rule-based executive logic, needed for overall system control and coordination of training activities. In the Phase I of this SBIR project, we developed a rulebase, message database, and control augmentation logic to train the fixed point hover maneuver. Under the Phase II SBIR effort described here, we developed additional training modules for hover taxi, hover turn, traffic pattern, and land from hover maneuvers. We extended the control augmentation logic into a fully-gain scheduled variable stability augmentation system, and significantly expanded the scope of the verbal feedback to provide procedural cueing on proper maneuver execution and coordination, in addition to corrective feedback on overt manual task performance.

Procedure:

We specified a Phase II effort focusing on the development of a full-scope Intelligent Flight Trainer for initial entry rotary-wing training. We conducted a four-task development, demonstration, and validation program:

- Development and implementation of PC-based system architecture
- Expansion of IFT scope
- Demonstration of operation
- Specification of requirements for full-scope IFT

Findings:

This Phase II effort led to the following major findings:

- **Demonstration of microprocessor-based hosting of simulation and training technologies in a distributed computing environment**: In Phase I the MicroVAX-based UH-1 TRS was linked with a PC-based IFT. During Phase II we transitioned to an all-PC architecture, using a distributed computing environment that isolated all training functions on a single machine.

We developed and implemented a UH-1H simulator model for PC implementation, and interfaced it with authentic cockpit controls and a glass cockpit display system.

- **Expansion of IERW training scope**: We expanded the IFT's scope from training only the fixed-point hover maneuver to add hover taxi, hover turn, traffic pattern, land from hover, and takeoff to hover. We developed the ES rulebases, message sets, and stability augmentation logic to support the training of these maneuvers. In Phase I the message sets and expert system logic pertained primarily to the student's manual control performance; under Phase II we expanded the scope of the voice feedback to provide greater procedural cueing.
- **Successful demonstration of PC-based prototype**: We successfully installed and demonstrated a multi-PC system, and verified the operation its key components: adaptive aiding that varied flight task difficulty with student proficiency, expert system emulation of IP advice and student diagnosis, and synthetic voice feedback of this advice to the student. An informal training exercise with a neophyte student demonstrated the system's ability to help the student develop proficiency at the hover, hover turn, and traffic pattern maneuvers.

Recommendations:

This Phase II effort has enabled us to develop numerous insights pertaining to the design of intelligent flight training devices. We now summarize our major recommendations for transitioning the research prototype developed here into a fully functional training tool:

1) **Conduct a formal validation and transfer of training experiment**: The IFT's ability to teach students how to perform all of its maneuvers and to deliver positive transfer of training to the flight line must be tested for a meaningful sample of students.
2) **Transition to the TH-67 helicopter:** The current implementation and its VAX-based predecessor at the ARI Simulator Facility use a UH-1H vehicle model. Since current Army flight training curricula specify the use of the TH-67 it is appropriate to replace the UH-1H simulation model with one of the TH-67 of sufficient fidelity to meet training requirements.
3) **Develop training modules for conducting "air work"**: The IFT presently trains fixed-point hover, hover taxi, hover turn, traffic pattern, takeoff to hover, and land from hover maneuvers. These maneuvers were selected through consultation with the Contracting Officer's Technical Representative. There is a considerable increase in complexity and difficulty between the so-called "hover maneuvers" and the traffic pattern. In order to help students learn how to handle the helicopter during high-speed forward flight, it would be appropriate to practice short finite-duration maneuvers such as level flight, climbs, turns, descents, etc., before attemping a full traffic pattern.
4) **Install all system components on a single PC**: Since this Phase II effort was awarded, there has been a considerable improvement in the computing power of personal computers. Modern PCs based on the Pentium chip provide enough raw computing power to perform all IFT/SIM computational tasks at a sufficient speed. In the past year, numerous vendors have introduced low-cost textured image generator boards for PCs that can deliver multi-channel textured workstation-quality graphics at a fraction of the cost. Consolidation onto a single platform would drastically simplify the system and provide greater flexibility in specification of graphical content on the instrument panel and the external scene. For example, "highway-in-the-sky" displays might assist students in tracking a nominal approach path. Or, specific instruments on the glass cockpit display could be highlighted when the related variable is out of tolerance, to help the student focus his/her attention on that quantity. This could provide for a kind of "adaptive display augmentation," to complement the stability augmentation already provided. Such a system might provide additional visual cues in the out-of-window display and on the instrument panel at high help levels, and gradually remove these elements as the student demonstrates performance proficiency improvement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

The Army Research Institute (ARI) has developed and demonstrated the UH-1 Training Research Simulator (TRS), and shown it to be effective in providing low-cost, effective simulator-based basic training of initial-entry rotary wing (IERW) maneuvers. The system integrates technologies of distributed computational processing and computer image generators (CIGs) in a configuration that has the potential of minimizing dependence on a dedicated instructor pilot. Six empirical studies have demonstrated the system's effectiveness in delivering positive *transfer-of-training* (TOT) to the UH-1 helicopter (Dohme, 1990, 1991), and further studies are planned to expand the evaluation effort's scope.

A need has existed, however, to improve the instructional portions of the system by incorporating instructor pilot (IP) domain knowledge, and by providing the student pilot (SP) with appropriate tutorial feedback regarding proficiency progress. By doing so, significant savings in IP hours could be realized to complement the savings already achieved through the low-cost hardware configuration. Of perhaps greater significance was the potential of demonstrating how intelligent tutoring systems (ITSs) could be hybridized with flight training simulators to support the development of a new generation of low-cost *intelligent flight trainers* (IFTs). The potential thus existed far beyond the UH-1 TRS, and may have significant impact on several other existing and planned flight trainers.

In order to accomplish these objectives, we proposed that the IP's domain knowledge be encapsulated in the form of an expert system (ES), and that the entire system be hosted a PC-based distributed computing environment. Feedback would be provided to the student verbally via a synthetic voice generator that was already a component of the existing system. As we saw it, the ES/synthesizer combination could implement four basic functions directly supporting verbal feedback to the student pilot:

- A tutorial function to remind the student how to implement good perceptual or control strategies (e.g., "Remember to apply left pedal as you raise the collective.").
- A performance monitoring function for performance advisory messages (e.g., "You are too high.").
- A control activity monitoring function to provide feedback on the student's control usage (e.g., "You are thrashing the collective.").
- An advisory function that makes explicit the suggested control strategy or corrective maneuvering (e.g., "Slow down using aft cyclic.").

Furthermore, the presence of an expert system shell within the system could support the implementation of rule-based executive logic, needed for overall control of the system and coordination of training activities. In the Phase I of this SBIR project, we developed a rulebase, message database, and control augmentation logic to train the fixed point hover maneuver. Under the Phase II effort described in this report, we developed additional training modules for hover taxi, hover turn, traffic pattern, and land from hover maneuvers. We extended the control augmentation logic into a fully-gain scheduled variable stability augmentation system, and significantly expanded the scope of the verbal feedback to provide procedural cueing on proper maneuver execution and coordination, in addition to corrective feedback on overt manual task performance.

1.1 Technical Objectives and Approach

On the basis of our Phase I results, we specified a Phase II effort focusing on the development of a full-scope Intelligent Flight Trainer (IFT) for initial entry rotary-wing training. We conducted a four-task development, demonstration, and validation program:

- Development and implementation of PC-based system architecture
- Expansion of IFT scope
- Demonstration of operation
- Specification of requirements for full-scope IFT

We first **designed, developed, and implemented a PC-based system architecture** to exploit the advantages of parallel processing and high-speed Ethernet communication. The initial design for the PC-based architecture employed three computers: 1) the simulator model host; 2) the IFT; and 3) the cockpit interface processor (CIP), intended to provide the visual interface between the student in the vehicle cab and the simulator model/IFT. The simulator host contains the UH-1 equations of motion, which are updated at each time step using pilot inputs provided to the computer by the CIP, which samples pilot inputs at a 30 Hz rate. On each time step, the simulator host transmits the vehicle state and control position information to the IFT, which uses them to assess the student's performance at the intended maneuver. This assessment provides the basis for the IFT advisor's synthetic voice feedback (using text-to-speech software) to the student via a speech board. It was originally intended that the speech board would be hosted within the IFT machine. However, we discovered that the installed speech board was incapable of asynchronous operation (i.e., the computer could not "talk" to the student while performing other computational functions). In the interest of minimizing development time and costs, we chose to continue to use the same speech board installed on a fourth PC, which communicates with the

IFT using an asynchronous serial connection. The IFT also computes feedback control inputs for stability augmentation, which are sent back to the simulator host for input into the vehicle dynamics. After updating the vehicle state, the simulator host transmits instrument readings to the CIP, which displays them to the student on a glass cockpit display. Originally, our design had called upon using the LabView environment to perform all control sampling and instrument display functions. However, we discovered that LabView was not capable of maintaining a 30 Hz update rate with all of the necessary instruments. This necessitated the addition of a fifth computer to the system (a Silicon Graphics workstation) for actually drawing the cockpit instruments. The simulator host also transmits vehicle position and attitude data to the BBN 120-TX image generators, which produces the out-of-window display that the student sees in the cockpit cab. In our Phase II effort, the simulator host PC was made to communicate directly with the image generators using dedicated interface hardware.

Following successful initial implementation of the PC-hosted system, we focused our attention on **expanding the scope of the IFT's training modules**. In the Phase I effort the IFT's functionality was limited to training a fixed-point hover maneuver. In Phase II we expanded the IFT's scope to include hover taxi, hover turn, traffic pattern, land from hover, and takeoff to hover maneuvers. For each of these maneuvers, we reviewed Army training guidelines (U.S. Army, 1994) and conducted knowledge engineering exercises with the Contracting Officer's Technical Representative (COTR) and others at Fort Rucker to identify suitable strategies and message sets for training these maneuvers. These exercises provided the basis for developing CLIPS (Giarratano, 1993) expert system modules for training each of the maneuvers. We also extended the functionality of the helper control logic so that it could provide effective stability augmentation across forward flight speeds for a range of "help levels."

Upon completion of the substantive engineering development, we **demonstrated the system's operation** by conducting a set of exercises to evaluate the IFT's ability to train a neophyte student in the basic flight maneuvers. The student demonstrated sustained improved performance in the hover and hover turn maneuvers, and properly followed the IFT's procedural guidance during a traffic pattern. The results are encouraging, and they set the stage for a more formal evaluation of the IFT's training potential in a transfer of training experiment.

Finally, we developed specifications for the development of a full-scope intelligent flight trainer. In the course of this Phase II effort, we have developed considerable insights into the nature of the flight training problem and the potential for simulator-based training. We believe that these insights provide the foundation for transitioning the research prototype developed in this effort into a full-scope training device. We present our recommendations in Chapter 5.

1.2 Summary of Results

The primary result of the predecessor Phase I effort was the successful proof-of-concept demonstration of an intelligent flight trainer for initial entry rotary wing training. The Phase II design, development, and implementation effort led to the following major findings:

- **Demonstration of microprocessor-based hosting of simulation and training technologies in a distributed computing environment**: In Phase I the MicroVAX-based UH-1 TRS was linked with a PC-based IFT. During the Phase II effort we transitioned to an all-PC architecture, using a distributed computing environment that isolated all training functions on a single machine. We developed and implemented a UH-1H simulator model for PC implementation, and interfaced it with authentic cockpit controls and a glass cockpit display system.

- **Expansion of IERW training scope**: In the Phase I effort the IFT's functionality was limited to training a fixed-point hover maneuver. In Phase II we expanded the IFT's scope to include hover taxi, hover turn, traffic pattern, land from hover, and takeoff to hover maneuvers. We developed the ES rulebases, message sets, and stability augmentation logic to support the training of these maneuvers. In Phase I the message sets and expert system logic pertained primarily to the student's manual control performance; under Phase II we expanded the scope of the voice feedback to provide greater procedural cueing (e.g., for transitions from one flight condition to another).

- **Successful demonstration of PC-based prototype**: We successfully installed and demonstrated a multi-PC system, and verified the operation of the IFT's key components: adaptive aiding that varied flight task difficulty with student proficiency, expert system emulation of IP advice and student diagnosis, and synthetic voice feedback of this advice to the student. An informal training exercise with a neophyte student demonstrated the system's ability to help the student develop proficiency at the hover, hover turn, and traffic pattern maneuvers.

These findings established the basic feasibility of the proposed system, and provide the basis for a formal system evaluation in a transfer of training experiment. The study was specifically structured to be narrow in scope, but capable of generating the foundations for a fully functional trainer.

1.3 Report Outline

This report contains four additional chapters and six appendices.

Chapter 2 provides a literature review on the subject of intelligent tutoring and simulation-based training, and describes the enabling technologies needed to develop the IFT, namely: 1) the Optimal Control Model (OCM) for student pilot modeling, and 2) expert systems for implementation of intelligent tutoring.

Chapter 3 describes the design and development of the IFT. Section 3.1 provides a functional block diagram of the overall system. ection 3.2 describes the UH-1 simulation model. Section 3.3 describes the principal components of the IFT, which are 1) the expert system shell with voice feedback, and 2) the variable stability augmentation control law, designed to make the helicopter easier to control for neophyte students. Finally, section 3.4 describes the system's hardware implementation at Fort Rucker.

Chapter 4 presents the results of efforts to validate the system. Section 4.1 describes ongoing efforts to tune the UH-1 model, while section 4.2 presents a set of training results derived from experiments conducted during the Phase II effort.

Chapter 5 concludes the main body of the report with a summary, conclusions, and recommendations for follow-on development.

Appendix A provides a "User's Guide" to the IFT system installed at the Aviation R&D Facility at Fort Rucker, AL. Appendix B contains an overview of the IFT's software modules and components, and is intended for users planning to modify the IFT source code. Appendix C describes the control wiring in the IFT cockpit. Appendix D provides a description of LabView control calibration routines for the CIP. Appendix E describes the Ethernet communication protocol between the CIP and the SGI 4D/50, which actually presents the cockpit instrument readings to the student. Finally, Appendix F contains a number of photographs of the hardware setup at Fort Rucker.

2. Enabling Technologies for Phase II Development

Our Phase II effort relied on a number of key enabling technologies for system development, and we now describe them in detail. Section 2.1 provides an overview on the subject of intelligent tutoring systems. Section 2.2 discusses expert systems for rule-based behavior. Finally, section 2.3 provides an overview of the optimal control model for pilot modeling.

2.1 Intelligent Tutoring Systems

Intelligent tutoring systems seek to train a user in a computerized environment, and they are generally developed to take the place of a human instructor. Significant research has been conducted in ITS design and application, as indicated by a number of recent publications (Jones, Abbott & Burley, 1992; Kaplan, Trenholm, Gitomer & Steinberg, 1993; Kaplan & Rock, 1995; Orey & Nelson, 1993). Figure 3.1-1 illustrates the general architecture for a range of ITSs. Three primary models are involved (Steinberg, 1991): a student model, a teacher model, and a domain expert model. Within a given application, the student model changes with the student's progress, while the teacher and expert models typically remain fixed. With this type of modular format, domain expert models may be developed for various subjects and may be interchanged within the ITS without modifications to the teacher model. Additionally, teacher models may be specialized for specific domains and interchanged along with the expert model. For each student using the ITS, a separate dynamic model may be created to represent individual student progress.

In general, within the ITS, the teacher model will compare the student model with the domain expert model to determine the student's progress. From this information, the optimal teaching strategy to minimize the difference between the two models is determined. An appropriate task is then selected and presented to the student via the student/machine interface. Upon completion of the task, the ITS adds the student's response to the history of past results. This history is then used by the ITS to update the dynamic student model. The ITS continues the process until either the actual student or the teacher model is satisfied with the performance.

One example of teacher model/student model interaction involves the teacher model attempting to purposely evoke wrong answers from the student, in order to point out errors and illuminate solutions. To accomplish this, the teacher model presents the student model with various problems, thereby determining with which ones the student will have difficulty.

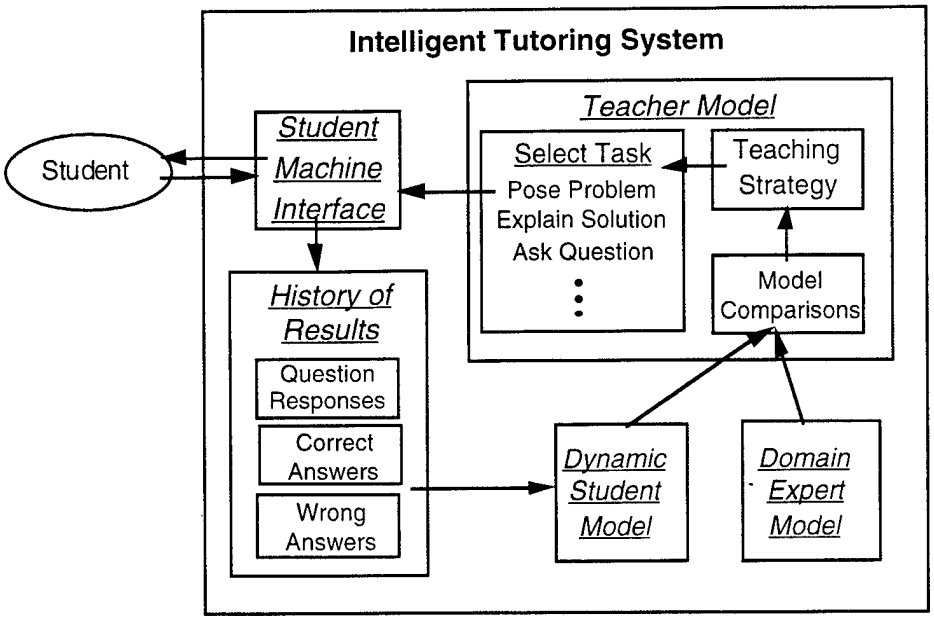


Figure 2.1-1: General Architecture of an Intelligent Tutoring System

Expert systems (ESs), discussed in greater detail in the following section, have been employed in ITS development for numerous applications. Goldstein (1982) built an expert model with a graph that defines the links between the rules that an expert uses. The teacher model (or tutor) attempts to bring the student to the expert level by watching the student's knowledge acquisition and then modeling the student's knowledge as an overlay on the expert's knowledge. Kimball (1982) modeled the student's knowledge by assigning probabilities to the likelihood that a student knows certain facts given that he knows other facts. The teacher model compares the student's response to an expert's response and chooses remedial work, if necessary. Burton (1982) defined right and wrong rules pertaining to subtraction, where wrong rules are the misconceptions people have about subtraction. With this expert model, he modeled each student's heuristics and built this tutor to recognize when the student uses incorrect rules and to present examples of where the incorrect heuristics fail.

Expert systems have also been used in ITSs for teaching electronic troubleshooting via SOPHIE (Brown, Burton & deKleer, 1982), identifying bugs in Pascal programs for novices (Johnson & Soloway, 1985), tutoring proofs in high school geometry (Anderson, Boyle & Reiser, 1985), teaching complex industrial process control strategies (Woolf, Blegen, Jansen & Verloop, 1986), and teaching simple fractions (Gutstein, 1992). Under a recent Phase I SBIR effort, we developed a prototype simulator-based ITS for training the Army medical specialist (Zacharias, Barros & Njie, 1994).

Recently, there has been greater interest in using ITSs to teach skills that make more perceptual or motor demands on the student. One particular example is the ITS designed for fixed-wing flight training (Regian, 1989). Here, an expert module monitors the student's flight proficiency (in simulation) over different maneuvers, and a coach (teacher model) advises the student using synthetic voice feedback; CLIPS (Giarratano, 1993) is used to implement the real-time ES.

2.2 Expert Systems for Rule-Based Behavior

An expert system (ES) is a computer program that can perform a task normally requiring the reasoning ability of a human expert. ESs are highly specialized according to their application domains. Although any program solving a particular problem may be considered to exhibit expert behavior, ESs are differentiated from other programs according to the manner in which the domain specific knowledge is structured, represented and processed to produce solutions. In particular, ES programs partition their knowledge into the following three blocks: Data Base, Rulebase and Inference Engine. ESs utilize symbolic and numeric reasoning in applying the rules in the Rulebase to the facts in the Data Base to reach conclusions according to the construct of reasoning specified by the Inference Engine.

There are two basic types of knowledge that can be incorporated into expert systems: declarative knowledge and procedural knowledge. The kind of knowledge describing the relationships among objects is called declarative knowledge. The kind of knowledge prescribing the sequences of actions that can be applied to this declarative knowledge is called procedural knowledge. In expert systems, procedural knowledge is represented by production rules whereas declarative knowledge is represented by frames and semantic networks, in addition to production rules. Rules are expressed as IF-THEN statements. When the IF portion of a rule is satisfied by the facts, the rule is fired by executing the statements specified by the THEN portion. Typically, the production rules deal with uncertainty through the use of certainty factors, probability or fuzzy logic.

The inference control strategy is the process of directing the symbolic search associated with the underlying type of knowledge represented in an expert system: antecedents of IF-THEN rules, nodes of a semantic net or a collection of frames. In practical expert system applications, the blind search is an unacceptable approach due to the associated combinatorial explosion. Search techniques can be basically grouped into three: breadth-first, depth-first and heuristic. The breadth-first search exhausts all nodes at a given level before going to the next level. In contrast, the depth-first exhausts all nodes in a given branch before backtracking to another branch at a

given level. Heuristic search incorporates general and domain-specific rules of thumb to constrain a search.

Expert systems employ basically two types of reasoning strategies based on the search techniques above: forward chaining and backward chaining. In forward chaining, starting from what is initially known, a chain of inferences is made until a solution is reached or determined to be unattainable. For instance, in rule-based systems, the inference engine matches the left-hand side of rules against the known facts, and executes the right-hand side of the rule that is activated. In contrast, backward-chaining systems start with a goal and searches for evidence to support that goal. Pure forward chaining is appropriate when there are multiple goal states and a single initial state whereas backward chaining is more appropriate when there is a single goal state and multiple initial facts. Many expert systems use both forward and backward chaining in a hybrid approach.

To provide even a rudimentary representation of the expertise of an instructor pilot requires a knowledge engineering effort. After an initial review of relevant training manuals (e.g., Army, 1994), this took the form of detailed structured interviews with several instructor pilot experts along the lines indicated in figure 2.2-1. Here we show a knowledge engineer converting the subject matter expert's knowledge into the facts and rules that make up the ES, via direct interview, procedural normalization, and knowledge base implementation.

In our Phase II effort, we conducted knowledge engineering exercises with domain experts at Fort Rucker to identify tutorial strategies for performing simulator-based training of hover, hover taxi, hover turn, traffic pattern, takeoff to hover, and land from hover maneuvers. In Chapter 3 we describe in greater detail the rulebases developed through this effort. The expert system modules were implemented using the CLIPS shell (Giarratano, 1993).

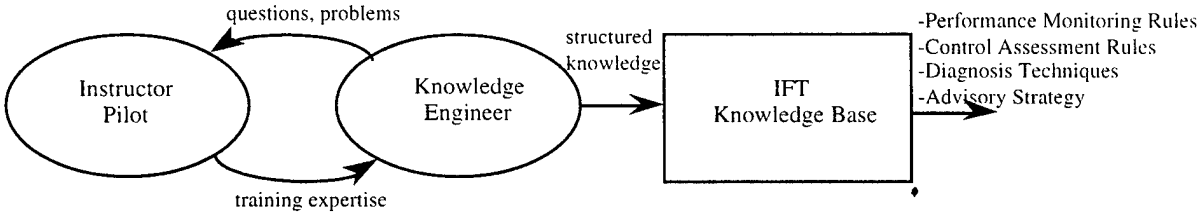


Figure 2.2-1: IFT Knowledge Engineering Process

2.3 Pilot Modeling using the Optimal Control Model

Our approach to modeling the student pilot under Phase I centered on the Optimal Control Model (OCM) of the human pilot, a model which has been developed within the systems framework of modern estimation (ME) and control theory (Kleinman, Baron & Levison, 1970).

The basic assumption underlying the model is that the well-trained, well-motivated human operator behaves optimally in some sense, subject to inherent psychophysical limitations which constrain the range of his behavior. In the flight control environment, the model is capable of predicting steady-state task performance (e.g., RMS gunsight tracking error), frequency-domain pilot transfer functions (e.g., stick response to a wind gust), frequency-domain pilot *remnant* (e.g., stick jitter), and time-domain dynamic histories (e.g., critical trajectory variables during a piloted *pop-up* maneuver).

A general block diagram of the OCM is given in figure 2.3-1; a detailed description may be found in (Kleinman et al., 1970) and an overview may be found in (Krishnakumar, Sawal, Bailey & Dohme, 1991). As shown, the model consists of the following:

- A set of vehicle dynamics which define the basic system states $\mathbf{x}$ to be controlled by the pilot. In the general flight scenario, these would be vehicle attitude and location in navigation coordinates, vehicle linear and angular velocities, as well as other variables influencing pilot/vehicle response (e.g., SAS dynamics, display delays, etc.).

- A display interface which converts system states $\mathbf{x}$ into displayed variables $\mathbf{y}$, seen by the pilot. In the cockpit, these might range from a simple digital indication of airspeed, to an abstracted set of pitch attitude bars, to a highly pictorial volumetric representation of instantaneous weapons envelope. In effect, the display interface converts implicit system states to explicit display variables.

- A perceptual model that converts these display variables $\mathbf{y}$ into noisy and time-delayed *perceived* variables, denoted by $\mathbf{y}_p$. As shown in the diagram, this is accomplished by the formal addition of *observation noise* to the display $\mathbf{y}$, followed by a time delay. The noise is included to account for the pilot's visual acuity limits, limitations in his attention-sharing capacity, (which is reflected as imprecision, or noise in perception), and variations in the pilot's response strategy (due to, say, fatigue or idiosyncratic behavior). The noise also accounts for limitations in the man-machine interface, due to such factors as resolution limits, quantization levels, etc. The time delay is included to account for human delays in processing the information available in the cockpit, as well as for inherent delays in the display system itself.

- An information-processor or *equalization* block, which converts these perceived variables $\mathbf{y}_p$ to commanded control actions $\mathbf{u}_c$. As shown, this consists of: an optimum (Kalman) estimator, which, in tandem with a predictor, generates a minimum variance estimate of the current system state $\mathbf{x}$, and a set of *optimal gains* which are chosen to ensure that the resulting command $\mathbf{u}_c$ will minimize a pre-defined cost function that expresses the task

requirements. In an air-to-ground engagement, the cost function could include a term proportional to the error between target LOS and weapon boresight; in an air-to-ground engagement, it could also reflect deviations from desired weapon lethality envelope.

- An *equivalent neuromotor model* that converts this command $\mathbf{u}_C$ into a piloted control action $\mathbf{u}$. As shown, this is accomplished by the formal addition of *motor noise* to the command $\mathbf{u}_C$, and limited-band-width filtering, which together account for the pilot's neuromotor bandwidth limitations, and his inability to generate perfectly precise control actions.

The OCM has been validated against experimental data for a variety of tasks, and detailed results may be found in the literature. In one of the earliest validation studies involving closed-loop tracking, it was found that error scores, describing function, and operator randomness were affected by multiple-task requirements all in the manner predicted by the model (Levison, Baron & Kleinman, 1969). Gai & Curry (1976) used the OCM information-processing structure to analyze failure detection in a simple laboratory task, and in an experiment simulating pilot monitoring of an automatic approach. They reported good agreement between predicted and observed detection times for both abstract and realistic situations.

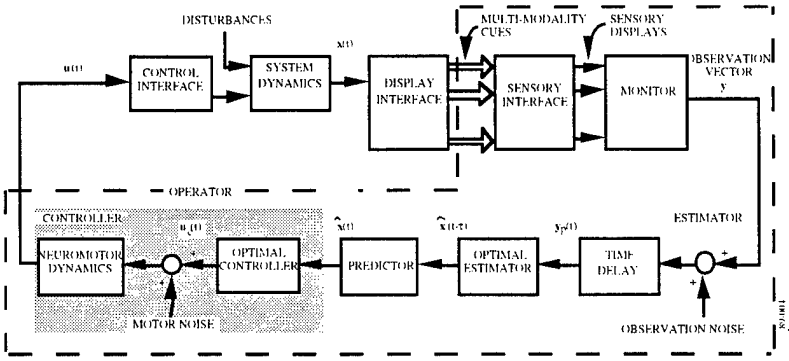


Figure 2.3-1: Optimal Control Model of Pilot/Vehicle System

Several studies have used the OCM in realistic workload environments to evaluate pilot behavior. Thus, aircraft display requirements have been investigated via model analysis by Kleinman & Baron (1971), and a model-based instrument display design procedure was developed by Hess (1977a). Hess (1976) analyzed system design effects on flight performance in a terminal configured vehicle (TCV) via the model, and anti-aircraft artillery (AAA) tracking has been similarly modeled, by Kleinman & Killingsworth (1974). He used the model to predict pilot performance in the flare and touchdown phase of STOL landing. More recently, we have used the model to evaluate the impact of different display designs on terrain-following performance (Zacharias, 1985; Zacharias & Brun, 1986; Gonsalves, Kneller & Zacharias, 1989b). In short, the

OCM has been applied in a number of varied non-laboratory situations, and has provided researchers with a common structure for understanding human performance.

In section 3.3.2 we describe how the OCM is used to implement what effectively is a variable stability augmentation control law in the IFT, to make the subjective difficulty of flying the aircraft dependent on student performance.

3 Intelligent Flight Trainer System Design

3.1 Functional Block Diagram

Figure 3.1-1 presents a functional block diagram of the simulator-based intelligent flight trainer developed during this effort. The IFT is shown as an "add-on" to a pilot-in-the-loop simulation. As shown, the IFT provides two key functionalities: 1) a *helper* control system, and 2) an expert system *advisor*. The intent of the helper is to make the vehicle easier to control through the use of stability augmentation. The IFT's performance evaluator dictates the help level (i.e., stability augmentation level) to the control system as a function of demonstrated student performance. As the student's performance improves, the help level decreases. Conversely, as student performance deteriorates, help level increases to further stabilize the vehicle. The performance evaluator, which is controlled by the overall ITS executive logic, uses pre-defined maneuver criteria (e.g., allowable lateral position deviations in a hover) to judge the student's performance. The ITS executive also controls the expert system advisor, which uses the performance evaluation, its own rulebase, and a message database to generate synthetic voice feedback that is sent directly to the student via an audio system. The advisor critiques the student's performance and provides corrective verbal feedback on errors that are out of tolerance.

A graphical user interface (which in the present implementation is a text-only display) provides the instructor or supervisor with control over the IFT's functions. The instructor uses this display to select training maneuvers and options (based on a desired training syllabus), and to monitor student progress during flight.

In the following two sections we describe the principal elements of the architecture presented in figure 3.1-1. Section 3.2 describes the UH-1H simulation model, and section 3.3 describes the intelligent flight trainer. Section 3.4 describes the overall system's hardware implementation, the cockpit interface, etc.

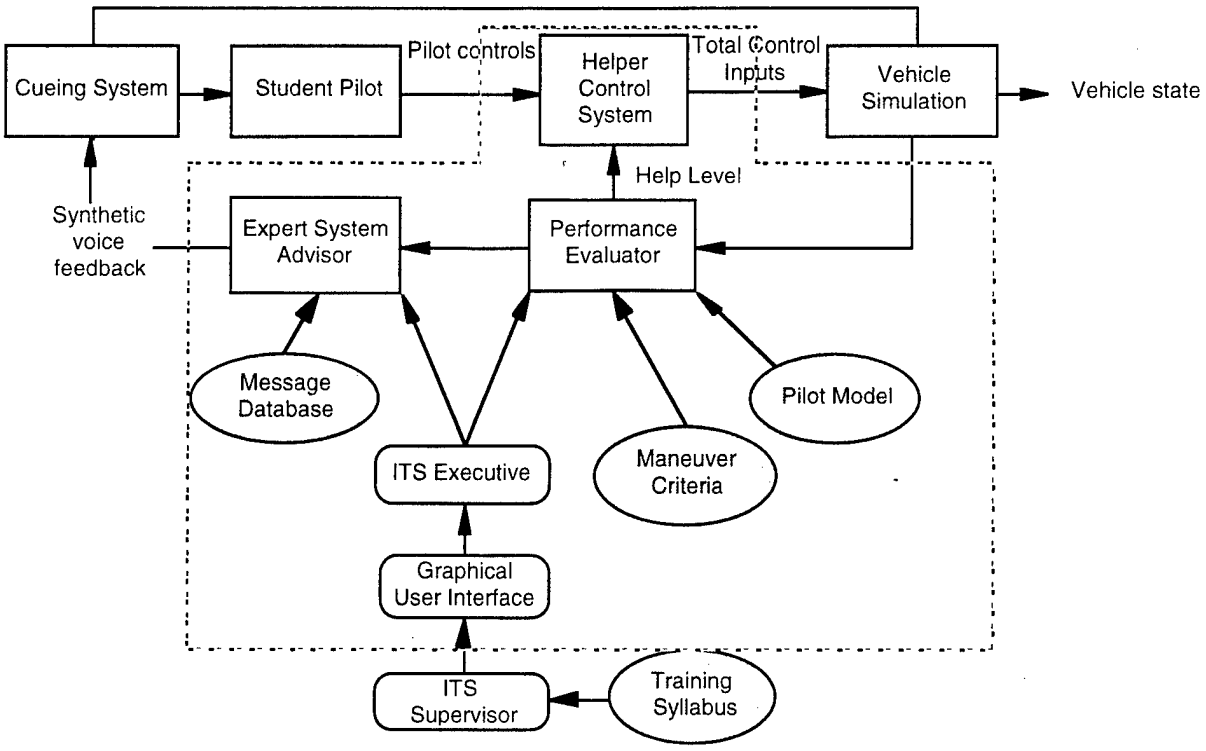


Figure 3.1-1: Functional Block Diagram of Simulator-Based Intelligent Flight Trainer

3.2 Simulator Model

The simulator host models the 6 degree-of-freedom dynamics of a Bell UH-1 (Huey) helicopter. This model, developed at the University of Alabama Flight Dynamics Laboratory, is based on the nonlinear *ARMCOP* model of the UH-1 (Talbot, Tinling, Decker & Chen, 1982). For application in the IFT, the original nonlinear model was linearized about a straight and level flight condition for 13 airspeed conditions, from -40 ft/sec to 200 ft/sec in intervals of 20 ft/sec. The resultant linearized aerodynamic matrices are used in conjunction with nonlinear kinematic and Euler angle equations. The vehicle state and control inputs are defined as

$$\mathbf{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & u & v & w & p & q & r & x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^T \quad (3.2\text{-}1)$$

$$\mathbf{u} = \begin{bmatrix} \delta_{lon} & \delta_{lat} & \delta_{col} & \delta_{ped} \end{bmatrix}^T \quad (3.2\text{-}2)$$

The state elements x_1, ..., x_5 model the dynamics of the helicopter's main rotor and stabilizer bar. The development and validation of this model is described extensively by Bailey, Krishnakumar, Whorton & Suman (1988). A number of modifications have been made to the basic vehicle model during this Phase II effort in order to improve its suitability for training low-altitude IERW maneuvers. Specifically, representations of ground effect, ground contact, and

effective translational lift (ETL) have been added. The engine-governor dynamics contained within the original ARMCOP model have been incorporated into the current model, to produce the signals needed to drive torque pressure and engine RPM indicators on the instrument display.

A set of operating instructions on the SIM software is provided in Appendix A. Appendix B describes the SIM source code and the compilation procedures in detail.

3.3 The Intelligent Flight Trainer

The IFT consists of two main computational modules:

- An expert system **advisor** that incorporates instructor pilot domain knowledge to provide the student with appropriate tutorial feedback regarding proficiency progress and corrective piloting techniques.
- An adaptive **helper** that adds inner-loop stability to the vehicle dynamics to make it easier for the student pilot to control the vehicle. The helper sets its augmentation level (i.e., feedback gains) as a function of student proficiency: when a student is performing poorly, feedback gains increase to further stabilize the vehicle. As the student's performance improves, the feedback gains decrease. The student's performance in a maneuver is judged satisfactory once he/she can maintain flight parameters within acceptable bounds for a minimum length of time without any stability augmentation from the helper.

Both of these components are now described, along with a discussion on the maneuvers that the IFT trains.

3.3.1 Expert System Advisor

The instructor pilot domain knowledge is implemented in the form of a CLIPS (Giarratano, 1983) rulebase. CLIPS is an expert system tool developed at NASA Johnson Space Center that facilitates development of software to model human knowledge or expertise. The expert system provides feedback to the student pilot (SP) directly through a synthetic voice generator, and it implements four basic functions:

- A tutorial function to remind the student how to implement good perceptual or control strategies (e.g., "Remember to apply left pedal as you raise the collective.").
- A performance monitoring function for performance advisory messages (e.g., "You are too high.").
- A control activity monitoring function to provide feedback on the student's control usage (e.g., "You are thrashing the collective.").

- An advisory function that makes explicit the suggested control strategy or corrective maneuvering (e.g., "Slow down using aft cyclic.").

The expert system also controls assignment of the helper's augmentation level by measuring student performance at the assigned task, and notifies the student of changes in help level (along with a description of the reason for the help level change). In the Phase I of this SBIR contract, we developed a rulebase and message database to train the fixed point hover maneuver. Under Phase II, modules were developed for hover taxi, hover turn, traffic pattern, and land from hover. Each of these maneuvers is now described.

3.3.1.1 Fixed-Point Hover

During a fixed-point hover, the student must maintain the helicopter in a fixed position over a specified location on the runway, with the nose parallel to the runway centerline. Rules and messages were developed to monitor the student's x, y, z, and ψ errors. The following performance indices define a quantitative measure of the student's overall longitudinal and lateral performance during the hover maneuver:

$$J_{lon} = k_x(x - x_{com})^2 + k_z(z - z_{com})^2 \tag{3.3-1}$$

$$J_{lat} = k_y(y - y_{com})^2 + k_\psi(\psi - \psi_{com})^2 \tag{3.3-2}$$

where x_{com}, y_{com}, z_{com}, and ψ_{com} are the desired values of x, y, z, and ψ respectively. The coefficients k_x, k_y, k_z, and k_ψ are constants that weight the contribution of each term to the overall performance index. They are defined as the reciprocal of the square of the allowable deviation of the related variable:

$$k_x = \frac{1}{\Delta x_{max}^2} \tag{3.3-3}$$

$$k_y = \frac{1}{\Delta y_{max}^2} \tag{3.3-4}$$

$$k_z = \frac{1}{\Delta z_{max}^2} \tag{3.3-5}$$

$$k_\psi = \frac{1}{\Delta \psi_{max}^2} \tag{3.3-6}$$

where Δx_{max}, Δy_{max}, Δz_{max}, and $\Delta \psi_{max}$ are nominally defined in Table 3.3-1. These thresholds were developed through consultation with the C.O.T.R. and Army flight training guidelines. Two thresholds J_{min} and J_{max} define what constitutes acceptable performance: if the student maintains

$J_{lon} < J_{min}$ and $J_{lat} < J_{min}$ for a specified time interval, IFT help level decreases. If either $J_{lon} > J_{max}$ or $J_{lat} > J_{max}$ for the same interval, help level increases.

Table 3.3-1: Flight Parameter Thresholds During Hover

Parameter	Value
Δx_{max}	±25 ft
Δy_{max}	±5 ft
Δz_{max}	±3 ft
$\Delta \psi_{max}$	±10°

When help level changes, the IFT alerts the student to the change and its cause. In the event of a help level increase, the IFT identifies the largest contributor to the performance index I as the "cause" of the help level increase (e.g., "You are too high. Help level is increasing."). The IFT judges the student's performance as satisfactory when he/she can maintain $J_{lon} < J_{min}$ and $J_{lat} < J_{min}$ without any stability augmentation (i.e., zero help) for a specified continuous time interval.

For each type of error, multiple levels of messages were developed, each more specific than the last. For example, when the helicopter's altitude drifts out of tolerance (as defined by Table 3.3-1), the student is advised to check the altitude. If the student does not correct the error within a certain length of time (10 sec in this case), the IFT tells the student that he/she is too low or too high. If the student is still unable to correct the error, the IFT tells the student what to do to correct the error and how to do it. After the highest message level is issued, the IFT resets the counter back to the first level. Table 3.3-2 lists all of the messages used during the hover maneuver.

The IFT prioritizes flight errors in accordance with the relevance of the related variable to flight safety. For the hover maneuver, this nominal prioritization is (in decreasing order of importance): z, x, y, and ψ. Thus if the student is out of tolerance on both x and y, the IFT first alerts the student to the x error. During validation of the hover trainer, it was realized that a fixed prioritization of flight errors was not truly reflective of instructor pilot strategy. For example, a human instructor pilot would not tell a student that the helicopter was a foot too high if it were also 100 ft off the side of the runway. This observation motivated the development of a *dynamic salience* strategy, which re-prioritized the rules if one of the flight errors was much larger than the rest. For small to moderate errors, the nominal prioritization was maintained. In subsequent evaluations, test subjects and IP observers found the resultant advisory strategy to be much more appropriate.

Table 3.3-2: Message Set for the Hover Maneuver

Error	Message Level	Message Text
x	1 (both)	Check your location over the runway.
	2 (forward)	You are too far forward
	2 (back)	You are too far back
	3 (forward)	You are too far forward; move back.
	3 (back)	You are too far back; move forward.
	4 (forward)	Move back using aft cyclic.
	4 (b ack)	Move forward using forward cyclic.
y	1 (both)	Check your location over the runway.
	2 (left)	You are too far left.
	2 (right)	You are too far right.
	3 (left)	You are too far left; move right.
	3 (right)	You are too far right; move left.
	4 (left)	Move right using right cyclic.
	4 (right)	Move left using left cyclic.
z	1 (both)	Check altitude.
	2 (both)	Check altitude. The horizon line should be about level with truck windows.
	3 (high)	You are too high.
	3 (low)	You are too low.
	4 (high)	You are too high; descend a bit.
	4 (low)	You are too low; climb a bit.
	5 (high)	Descend a bit using down collective.
	5 (low)	Climb a bit using up collective.
ψ	1 (both)	Check where your nose is pointing.
	2 (left)	Your nose is too far left of the centerline.
	2 (right)	Your nose is too far right of the centerline.
	3 (left)	Your nose is too far left of the centerline; turn it right.
	3 (right)	Your nose is too far right of the centerline; turn it left.
	4 (left)	Turn your nose right using right pedal.
	4 (right)	Turn your nose left using left pedal.

3.3.1.2 Hover Taxi

The student begins the hover taxi maneuver in a fixed-point hover at one end of the runway. The student must then accelerate and taxi the helicopter down the runway centerline at a speed of 4.5 knots, while maintaining lateral position and heading alignment over the runway centerline. At the end of the runway, the IFT instructs the student to slow down to a hover and land the helicopter safely. Section 3.3.1.5 describes the scoring logic for the landing. During the steady

segment of the maneuver, the performance indices are based on the helicopter's speed, altitude, lateral deviation from the runway centerline, and heading, as follows:

$$J_{lon} = k_V (V - V_{com})^2 + k_z (z - z_{com})^2 \tag{3.3-7}$$

$$J_{lat} = k_y (y - y_{com})^2 + k_\psi (\psi - \psi_{com})^2 \tag{3.3-8}$$

The coefficient k_V is defined as

$$k_V = \frac{1}{\Delta V_{\max}{}^2} \tag{3.3-9}$$

and the other coefficients are defined as in eqs. 3.4-4 to 3.4-6. Table 3.3-3 lists the allowable errors during the hover taxi, while Table 3.3-4 lists the hover taxi messages.

When the helicopter reaches the end of the runway, the IFT instructs the student to slow down and hover the helicopter behind a "Maltese Cross" at the end of the runway. The IFT then uses the hover rules and messages to help the student establish a stable hover. Once the student maintains all hover parameters within tolerance for at least 5 consecutive seconds, the IFT instructs the student to descend gradually and land. Section 3.3.1.5 describes the rules governing the landing maneuver.

Table 3.3-3: Flight Parameter Thresholds During Hover Taxi

Parameter	Value
ΔV_{max}	±3 ft/sec
Δy_{max}	±5 ft
Δz_{max}	±3 ft
$\Delta \psi_{max}$	±10°

Table 3.3-4: Message Set for the Hover Taxi

Error	Message Level	Message Text
V	1 (both)	Check your taxi speed.
	2 (fast)	You are taxiing too fast. Your speed should be that of a brisk walk.
	2 (slow)	You are taxiing too slow. Your speed should be that of a brisk walk.
	3 (fast)	Reduce your taxi speed.
	3 (slow)	Increase your taxi speed.
	4 (fast)	Reduce your taxi speed using aft cyclic.
	4 (slow)	Increase your taxi speed using forward cyclic.
y	1 (both)	Check position over centerline.
	2 (left)	You are too far left of the centerline.
	2 (right)	You are too far right of the centerline.
	3 (left)	Slide right towards the centerline while maintaining forward speed.
	3 (right)	Slide left towards the centerline while maintaining forward speed.
	4 (left)	Slide right using right cyclic.
	4 (right)	Slide left using left cyclic.
z	1 (both)	Check altitude.
	2 (high)	You are too high.
	2 (low)	You are too low.
	3 (high)	You are too high; descend a bit.
	3 (low)	You are too low; climb a bit.
	4 (high)	Descend a bit using down collective.
	4 (low)	Climb a bit using up collective.
ψ	1 (both)	Check where your nose is pointing.
	2 (left)	Your nose is too far left of the centerline.
	2 (right)	Your nose is too far right of the centerline.
	3 (left)	Your nose is too far left of the centerline; turn it right.
	3 (right)	Your nose is too far right of the centerline; turn it left.
	4 (left)	Turn your nose right using right pedal.
	4 (right)	Turn your nose left using left pedal.

3.3.1.3 Hover Turn

In the hover turn maneuver, the student is required to make alternating right and left 90° turns, while maintaining fixed x, y, and z coordinates. The student begins the maneuver facing north (from the south end of the runway), and first initiates a 90° turn to the East. During the turning segments the IFT expects the student to maintain a turn rate of 3°/sec. At the end of each turn segment, the student is instructed to establish a stable hover and then reverse the turn

direction. During the steady segment of the maneuver, the performance indices are based on the helicopter's turn rate, altitude, horizontal position, and heading, as follows:

$$J_{lon} = k_x(x - x_{com})^2 + k_z(z - z_{com})^2 \tag{3.3-10}$$

$$J_{lat} = k_y(y - y_{com})^2 + k_{\dot{\psi}}(\dot{\psi} - \dot{\psi}_{com})^2 \tag{3.3-11}$$

The coefficient $k_{\dot{\psi}}$ is defined as

$$k_{\dot{\psi}} = \frac{1}{\Delta\dot{\psi}_{\max}{}^2} \tag{3.3-12}$$

and the other coefficients are defined as in eqs. 3.4-4 to 3.4-6. Table 3.3-5 lists the allowable errors during the hover turn.

Table 3.3-5: Flight Parameter Thresholds During Hover Turn

Parameter	Value
Δx_{max}	±15 ft
Δy_{max}	±15 ft
Δz_{max}	±3 ft
$\Delta\psi_{max}$	±3°/sec

Due to the relatively short duration of each turn segment (at a 3°/sec average turn rate, a 90° turn will take only 30 sec), it was necessary to devise a means of minimizing the level of verbiage that the IFT produced. In the hover and hover taxi maneuvers, help level changes (and announcements thereof) were made as soon as the student met the criterion on the performance indices. This technique was modified for the hover turn because of the likelihood that the student might achieve a help level change just as he/she was approaching the end of a turn segment. If the IFT then informed the student of the help level change, the instruction to hover and then reverse the turn direction would be delayed by as much as 10 sec, by which point the student would overshoot the final heading angle. To alleviate this potential problem, the rulebase was structured so as to only *announce* help level changes when the student begins a north → east turn segment. The help level changes themselves are made in accordance with the previously defined logic; when the student starts a north → east turn, the IFT verbally informs the student of any help level changes since the last north → east turn. This particular moment was chosen because it was identified as a relatively non-time-critical instant of the maneuver.

The messages and rule logic pertaining to horizontal drift were also modified for this maneuver. When the helicopter drifts away from the nominal "pivot point" for the turn, the correctional advice is more complicated than in the case of the fixed point hover. For example, if

the helicopter's y drift exceeds the allowable tolerance, the proper control action to return to the pivot point will depend on the heading angle ψ: If the helicopter is facing north, left cyclic must be applied to return to the pivot point. Conversely, if the helicopter is facing due east, aft cyclic must be applied. At intermediate angles, both aft and left cyclic may be needed. To address this problem, the IFT software computes the angle between the vehicle centerline and an imaginary line connecting the vehicle center of gravity to the nominal pivot point, shown as σ in Figure 3.3-1. The region around the helicopter is discretized into octants as shown, and the "octant number" that the angle σ falls within dictates the direction-specific advice that the IFT gives the student on how to return to the pivot point. Table 3.3-6 lists the direction-specific messages used in each octant for x/y errors, while Table 3.3-7 lists the messages used for z and $\dot{\psi}$ errors.

When the student meets the performance requirements for the hover turn and ends a turn facing North, the IFT instructs him/her to establish a hover and then land the helicopter.

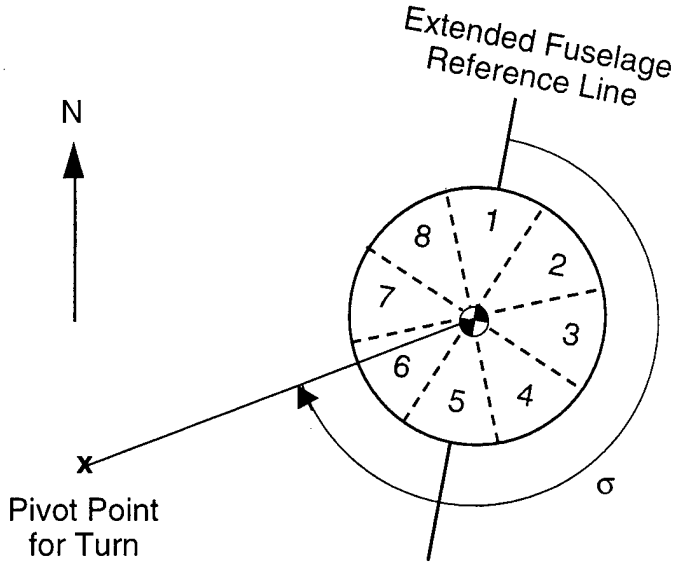


Figure 3.3-1: Computation of Relative Orientation between Helicopter and Pivot Point during Hover Turn.

Table 3.3-6: Direction-Specific Messages for Positional Errors during Hover Turn

Octant	Message
1	Return to pivot point using forward cyclic.
2	Return to pivot point using right and forward cyclic.
3	Return to pivot point using right cyclic.
4	Return to pivot point using right and aft cyclic.
5	Return to pivot point using aft cyclic.
6	Return to pivot point using left and aft cyclic.
7	Return to pivot point using left cyclic.
8	Return to pivot point using left and forward cyclic.

Table 3.3-7: Altitude and Turn Rate Messages for Hover Turn

Error	Message Level	Message Text
z	1 (both)	Check altitude.
	2 (high)	You are too high.
	2 (low)	You are too low.
	3 (high)	You are too high; descend a bit.
	3 (low)	You are too low; climb a bit.
	4 (high)	Descend a bit using down collective.
	4 (low)	Climb a bit using up collective.
$\dot{\psi}$ (left turns)	1 (fast/slow)	Check turn rate.
	2 (fast)	You are turning too fast.
	2 (slow)	You are turning too slow.
	3 (fast)	Reduce your turn rate.
	3 (slow)	Increase your turn rate.
	4 (fast)	Reduce your turn rate using right pedals.
	4 (slow)	Increase your turn rate using left pedals.
$\dot{\psi}$ (right turns)	1 (fast/slow)	Check turn rate.
	2 (fast)	You are turning too slow.
	2 (slow)	You are turning too fast.
	3 (fast)	Reduce your turn rate.
	3 (slow)	Increase your turn rate.
	4 (fast)	Reduce your turn rate using left pedals.
	4 (slow)	Increase your turn rate using right pedals.

3.3.1.4 Traffic Pattern

The traffic pattern is a rectangular flight pattern in the vicinity of the runway that begins and ends with a fixed-point hover at one end of the runway. Figure 3.3-2 presents a view from above.

The maneuver is considerably more difficult than the other maneuvers because the student must perform climbs, descents, turns, accelerations, and decelerations. Accordingly, the elements of the performance index depend on the segment of the maneuver currently being executed. The IFT also instructs the student on when to make flight transitions; i.e., when to start or end a turn, when to terminate an ascent, etc. The entire maneuver is divided into 12 segments, which are now described in order. Table 3.3-8 lists the error thresholds used during the maneuver (depending on the segment). The performance indices are defined in the same manner as the maneuvers described earlier; however, in this maneuver, the components of the performance indices vary as a function of the flight segment.

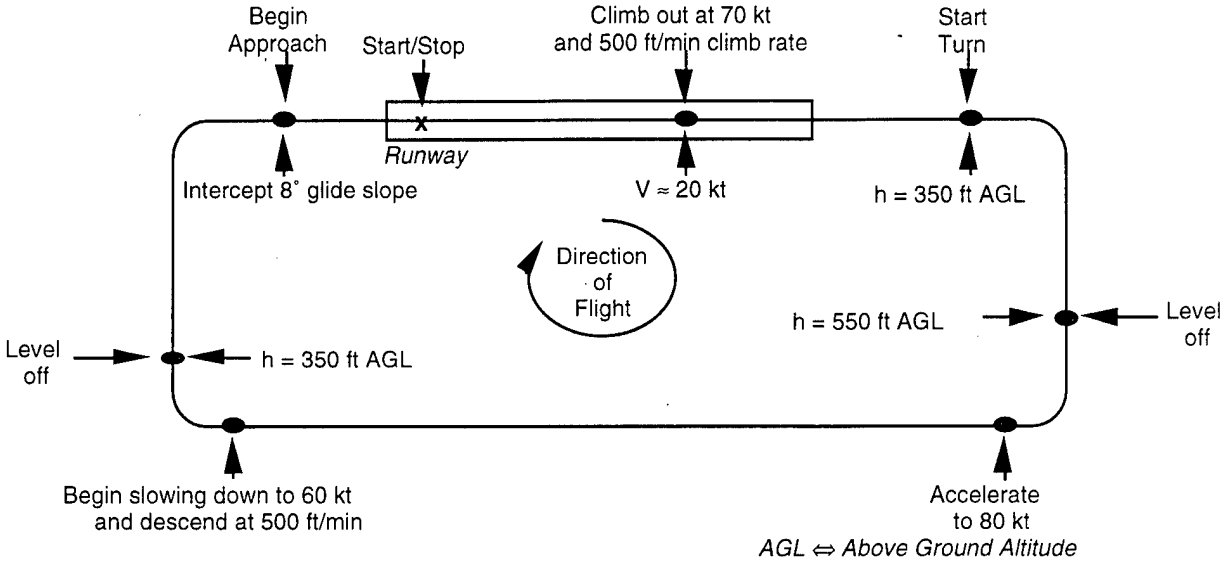


Figure 3.3-2: Schematic of Traffic Pattern

Table 3.3-8: Traffic Pattern Error Thresholds

Variable	Threshold
V	± 10 kt
y	± 100 ft
h	± 100 ft
$\dot{h}$	± 200 ft/min
ψ	± 10°
$\dot{\psi}$	± 1.5 deg/sec

1. Accelerate to Effective Translational Lift (ETL)

The traffic pattern begins with the helicopter in a stationary hover at the south end of the runway, facing north. The IFT instructs the student to accelerate down the runway centerline while maintaining constant altitude and lateral position over the centerline. The student should

not begin climbing until the helicopter reaches ETL, which occurs at an airspeed of approximately 20 kt.

2. Upwind Climb

Once the helicopter reaches ETL, the IFT instructs the student to adjust aircraft attitude to climb out at an airspeed of 60 kt and a climb rate of 500 ft/min. The student should maintain the original heading (north) and a ground track corresponding to the "extended" runway centerline.

3. Turn to Crosswind

The IFT instructs the student to begin a coordinated right turn to the crosswind leg once the altitude crosses 350 ft AGL. During the 3°/sec turn the student should maintain the airspeed at 60 kt and the climb rate at 500 ft/min. The student may reach the pattern altitude of 550 ft AGL during the turn, in which case he/she is instructed to level off while still turning. The IFT instructs the student to roll out of the turn once the heading is greater than 80°.

4. Crosswind

During the crosswind leg, the student must maintain airspeed at 60 knots and either continue climbing at 500 ft/min or maintain altitude at 550 ft AGL. In either case, the student should continue flying east, and maintain the easterly ground track that he/she established after rolling out of the turn.

5. Turn to Downwind

The length of the crosswind leg is set to 7,000 ft. As the student nears the end of the segment, he/she is instructed to initiate a coordinate right 90° turn to the downwind leg. During the turn, the student must either climb or maintain altitude, while maintaining a nominal 60 kt airspeed.

6. Downwind

When the student rolls the helicopter out the turn to downwind, the IFT directs the student to accelerate to an airspeed of 80 kt and either continue climbing or maintain the pattern altitude (550 ft AGL). During the downwind leg the student is required to maintain a heading of 180°.

7. Turn to Base

The downwind leg continues until the helicopter is approximately 5,000 ft past the south end of the runway (i.e., the same end of the runway at which the student began the maneuver). At this point, the student is instructed to begin a descending, decelerating, coordinated 90° right turn to the base leg of the maneuver.

8. Base

The student is expected to roll out of the turn to base facing west, at an altitude of 350 ft AGL and an airspeed of 60 kt. Since it is unlikely that a student trainee will be able to match these parameters exactly, the IFT informs the student of what action must be taken to match these parameters as he/she is rolling out of the turn; e.g., "Roll out of the turn heading west. Maintain airspeed at 60 knots and climb to 350 ft AGL." is the message given when the student rolls out of the turn more than 100 ft below the target altitude.

9. Turn to Final

When the student nears the end of the base leg, the IFT instructs the student to begin a right turn to the final leg, so that the helicopter may be lined up with the runway. During the turn, the student should maintain a turn rate of 3°/sec, an altitude of 350 ft AGL, and an airspeed of 60 kt.

10. Final

The objective of the final leg is to establish the approach lane; i.e., to line up the helicopter with the extended runway center line. During the final leg, the student is expected to maintain an airspeed of 60 kt at an altitude of 350 ft AGL.

11. Approach

The approach begins once the helicopter comes within 500 ft (of horizontal distance) of intersecting a 5° glide slope line that terminates at the final hover point. The student must begin decelerating and descending on the glide slope, while maintaining lane alignment and heading. The desired airspeed decreases linearly with range from the final hover point, so that the commanded airspeed reaches zero when the student reaches the terminal point.

12. Prepare to Land

This is the last segment of the traffic pattern. The student must establish a stable hover behind the "Maltese Cross" at the south end of the runway 5 ft above the ground, after which the IFT directs the student to land the helicopter.

3.3.1.5 Land from Hover

All of the maneuvers described in Sections 3.3.1.1 to 3.3.1.4 terminate with a landing from hover. The IFT judges the quality of an attempted landing using three sets of vehicle state measurements: 1) the vehicle state at the instant the IFT instructs the student to begin an approach, 2) the vehicle state at the instant of ground contact, and 3) the vehicle state at the moment when the vehicle comes to a complete stop. The distinction is made between the

moment of ground contact and the rest state to allow for the possibility that the student may skid the helicopter along the ground before coming to a full stop. Table 3.3-9 lists maximum allowable velocity components at the instant of touchdown, while Table 3.3-10 presents the maximum allowable positional deviations during an approach. The limits on forward speed $\dot{x}$ are asymmetric because backward motion is considered much more dangerous than slight forward motion at touchdown. The total positional deviations are computed during simulation by subtracting the vehicle state at rest on the ground from the vehicle state at the instant the descent to land commenced. The IFT also scores a key procedural activity during the maneuver: after the helicopter touches down, the student must lower the collective to the full-down position to receive a passing grade.

Table 3.3-9: Allowable Velocity Components at Touchdown

Parameter	Acceptable Range
$\dot{x}$	-1 ft/sec to 4 ft/sec
$\dot{y}$	-2 ft/sec to 2 ft/sect
$\dot{z}$	≥ -2 ft/sec
$\dot{\psi}$	-5°/sec to 5°/sec

Table 3.3-10: Allowable Total Drift during Land from Hover

Parameter	Maximum Deviation
x	±20 ft
y	±20 ft
ψ	±15°

3.3.1.6 Takeoff to Hover

The purpose of the takeoff to hover maneuver is to bring the helicopter from a rest condition on the ground to a stable, stationary hover. Much of the work of conducting a takeoff takes place on the ground, before the helicopter lifts off. As such, the rulebase governing the takeoff maneuver functions in great part to assist the student in preparing the vehicle for takeoff. Through consultation with the COTR and Army training guidelines, we partitioned the takeoff maneuver into three distinct segments and developed training rules for each:

1) **HEAVY**: The helicopter is on the ground, and the collective is below 1.5" of displacement. In this condition, the helicopter (that is, the simulation model of the helicopter) is incapable

of taking off or translating (or yawing) on the ground. At the commencement of the takeoff maneuver, the helicopter is in this condition. The IFT instructs the student to place the pedals and cyclic in near-neural positions, and gradually begin increasing the collective.

2) **LIGHT**: The helicopter is on the ground, and the collective is between 1.5" and 2.75" of displacement. Once the collective rises above this value, the helicopter is light enough on the skids that translational or yawing motion on the ground is possible. However, below 2.75" of displacement the helicopter is unable to leave the ground. The IFT provides the student with corrective verbal feedback to arrest all such motion. A maximum of ±3 ft/sec translational motion is allowed (lateral or forward/aft), and a maximum yaw rate of 1°/sec is permitted. The student must comply with these requirements before continuing to increase the collective.

3) **AIRBORNE**: The helicopter is in the air. If the student lifts the helicopter off the ground without bringing translational and rotational motion within specifications, the IFT judges the takeoff to be unacceptable and instructs the student to try again. If the conditions are met at the instant of takeoff, the IFT instructs the student to climb to a three-foot hover as soon as the helicopter is airborne. The helicopter must climb to the target altitude and maintain position within ±2 ft for the climb to be considered acceptable. Although the student must maintain altitude within ±1 ft in the real helicopter, we found that the lack of visual cues in the simulator made it very difficult to do so. As such, we expanded the threshold slightly.

Once the helicopter is airborne and all conditions have been met, the IFT automatically jumps into the next maneuver, which may be a hover, hover taxi, hover turn, or traffic pattern. If the student lifts the helicopter into the air and then descends and impacts the ground, the IFT judges the takeoff to be unacceptable. The student must then reset the system and try again.

The training logic for the takeoff maneuver is structured to help the student place the controls in such a position that when the helicopter finally lifts off from the ground, it will not exhibit any significant tendency to yaw or translate across the ground. Thus, in a properly executed takeoff the IFT will help the student conduct a smooth, vertical climb to the desired hover altitude.

3.3.2 Variable Stability Augmentation

The IFT provides adaptive training through the use of a student pilot *helper* . The helper provides inner loop stability augmentation, making it easier for the student to control the vehicle and conduct the desired maneuver. The use of this stability augmentation system for a fixed-point hover maneuver has previously been described (Krishnakumar et al., 1991; Zacharias *et al*;

1993). The helper strategy was selected so that the combined student and helper behaved as if the vehicle were under the control of a *skilled* pilot, as shown in Fig. 3.3-3.

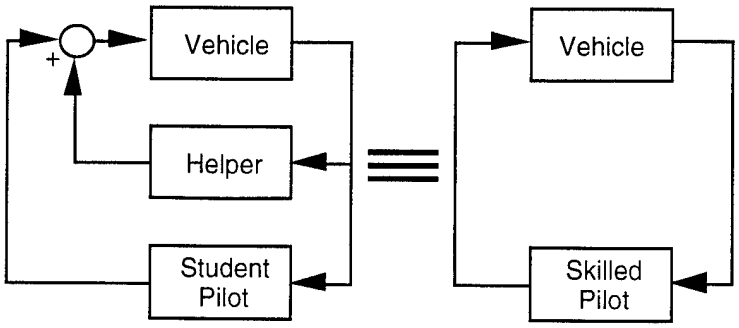


Figure 3.3-3: Student Pilot Helper Augmentation for Skilled Pilot Equivalence

The helper strategy was developed by modeling the student pilot and the expert pilot using the *Optimal Control Model* (OCM) (Kleinman et al., 1970), which is a control-theoretic representation of a human operator. Although previous studies (Zacharias & Brun, 1987) have shown that proficiency differences can be represented in a variety of ways within the OCM context, it was assumed here that the major difference between a student pilot and a skilled pilot was the performance index (PI) that each was attempting to minimize, given by

$$J = E\left[\lim_{T\to\infty}\frac{1}{T}\int_0^T\left(\mathbf{x}^T\mathbf{Q}\mathbf{x}+\mathbf{u}^T\mathbf{R}\mathbf{u}+\dot{\mathbf{u}}^T\mathbf{S}\dot{\mathbf{u}}\right)dt\right] \tag{3.3-1}$$

The pilot controls a dynamic system modeled by the system of ordinary differential equations,

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x}+\mathbf{G}\mathbf{u}+\mathbf{w} \tag{3.3-2}$$

where $\mathbf{x}$ is the vehicle state, $\mathbf{u}$ is the pilot control input, and $\mathbf{w}$ is a zero-mean Gaussian external disturbance input. The weighting matrices $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ determine the relative contribution of each component of $\mathbf{x}$, $\mathbf{u}$, and $\dot{\mathbf{u}}$ to the performance index J. For an expert pilot, the elements of these matrices are relatively large, since the expert pilot will place great emphasis on minimizing state deviations from the nominal conditions using small and smooth control deflections. By contrast, these matrix elements will be much smaller for a student pilot, owing to the student's inability to maintain small state deviations with smooth, low amplitude control inputs. Different levels of student proficiency are modeled using multiple sets of $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ matrices. The OCM formulation leads to a feedback control law of the form

$$\dot{\mathbf{u}} = -\mathbf{C}_1\mathbf{u}-\mathbf{C}_2\mathbf{x} \tag{3.3-3}$$

where the matrix $\mathbf{C}_1$ models the pilot's neuromuscular dynamics and $\mathbf{C}_2$ is the pilot's feedback gain matrix.

In the IFT, feedback gains were developed for a skilled pilot model and for 20 discrete student models (representing progressively increasing proficiency). Helper gains were then computed by subtracting the student model gains from the skilled pilot gains, as implied by Fig. 3.3-3. During operation, control gains were set as a function of student performance: as the student's measured performance improved, the augmentation level decreased.

The helper system developed in this manner for the fixed-point hover was found to be inappropriate for a complex flight maneuver such as the traffic pattern. The gain sets were developed using a model of the UH-1 linearized about hover, and were therefore not suited to providing stability augmentation at varying forward flight speeds. By design, the hover helper produced control actions that tended to oppose any attempt by the student to accelerate the helicopter. This feature was not suitable for the traffic pattern maneuver, in which the student would be required to perform accelerations, decelerations, climbs, descents, and turns (i.e., command time-varying *set points*). Finally, it was found that the procedure of subtracting one set of control gains from another to arrive at a helper model yielded highly oscillatory (and sometimes non-minimum-phase) step input responses at forward flight speeds. It was expected that such response characteristics would result in unpredictable handling qualities and only make the task of controlling the helicopter more difficult for the student.

In short, the more extensive training requirements made it necessary to develop a control architecture that can give the student authority over all axes of control and provide varying levels of stability augmentation (as well as acceptable handling qualities) across a full range of flight conditions. These considerations motivated the development of a gain-scheduled *non-zero set point regulator* (Stengel, 1986). At each discrete augmentation level, control gains are scheduled as a function of flight speed. Like the original OCM-based hover helper, the gains are determined by solving the Riccati equation associated with an LQR problem. However, instead of computing the helper gains by subtracting gains associated with a student model from those associated with a skilled pilot model, the helper gains are solved directly through appropriate definition of the LQR problem.

As before, the gain-scheduled helper provides 20 discrete levels of stability augmentation, but it now does this across a range of helicopter flight speeds. Control gains are designed at 9 forward flight speeds between 0 ft/sec and 160 ft/sec, in steps of 20 ft/sec. At the lowest augmentation level (i.e., zero help), the helicopter's response to control inputs is identical to its open-loop response and the control system provides no stability augmentation. At the highest

help level, the control response is well damped with minimal overshoot, qualitatively similar to the response that a skilled pilot flying an unaugmented vehicle might produce. In order to simplify the control design, it is assumed that the longitudinal and lateral-directional vehicle responses can be decoupled, and that controllers can be synthesized for each subsystem independently. Both axis designs are now described, following a description of the non-zero set point regulator.

3.3.2.1 Non-Zero Set Point Regulator Formulation

The helicopter motion equations are of the general form

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t)] \tag{3.3-4}$$

$$\mathbf{y}(t) = \mathbf{h}[\mathbf{x}(t), \mathbf{u}(t)] \tag{3.3-5}$$

where $\mathbf{x}$ is the vehicle state, $\mathbf{u}$ is the pilot control input, and $\mathbf{y}$ is a set of system outputs of interest. The state and control variables may be expressed as the sums of *nominal* and *perturbation* values:

$$\mathbf{x}(t) = \mathbf{x}_o(t) + \Delta\mathbf{x}(t) \tag{3.3-6}$$

$$\mathbf{u}(t) = \mathbf{u}_o(t) + \Delta\mathbf{u}(t) \tag{3.3-7}$$

Small perturbations from a nominal trajectory defined by $\{\mathbf{x}_0(t), \mathbf{u}_0(t)\}$ may be described using a linear approximation (Stengel, 1986). Expanding eq. 3.4-4 in a Taylor series and neglecting terms beyond the first degree, one obtains

$$\dot{\mathbf{x}}_o + \Delta\dot{\mathbf{x}} \cong \mathbf{f}(\mathbf{x}_o, \mathbf{u}_o) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Delta\mathbf{x} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\Delta\mathbf{u} \tag{3.3-8}$$

The nominal dynamic equations are

$$\dot{\mathbf{x}}_o(t) = \mathbf{f}[\mathbf{x}_o(t), \mathbf{u}_o(t)] \tag{3.3-9}$$

$$\mathbf{y}_o(t) = \mathbf{h}[\mathbf{x}_o(t), \mathbf{u}_o(t)] \tag{3.3-10}$$

The associated linear, time-varying linear *perturbation* equations are

$$\Delta\dot{\mathbf{x}}(t) = \mathbf{F}\Delta\mathbf{x}(t) + \mathbf{G}\Delta\mathbf{u}(t) \tag{3.3-11}$$

$$\Delta\mathbf{y}(t) = \mathbf{H}_x\Delta\mathbf{x}(t) + \mathbf{H}_u\Delta\mathbf{u}(t) \tag{3.3-12}$$

where

$$\mathbf{F}(t) = \frac{\partial \mathbf{f}(t)}{\partial \mathbf{x}}, \quad \mathbf{G}(t) = \frac{\partial \mathbf{f}(t)}{\partial \mathbf{u}} \tag{3.3-13a, b}$$

$$\mathbf{H}_x(t) = \frac{\partial \mathbf{h}(t)}{\partial \mathbf{x}}, \quad \mathbf{H}_u(t) = \frac{\partial \mathbf{h}(t)}{\partial \mathbf{u}} \tag{3.3-14a, b}$$

all of which are evaluated with $\mathbf{x} = \mathbf{x}_0(t)$ and $\mathbf{u} = \mathbf{u}_0(t)$.

The purpose of a linear quadratic regulator is to drive a linear, time-invariant system to a zero or non-zero set point optimally. A zero set point is one that lies on a solution to the nonlinear dynamic equations (i.e., $\mathbf{x}_0$, $\mathbf{u}_0$), and a non-zero set point is one possessing some steady state deviation $\{\Delta\mathbf{x}^*, \Delta\mathbf{u}^*\}$ from a nominal solution. It is useful to define the following relative variables for the subsequent development:

$$\Delta\tilde{\mathbf{x}}(t) = \Delta\mathbf{x}(t) - \Delta\mathbf{x}^* \tag{3.3-15}$$

$$\Delta\tilde{\mathbf{u}}(t) = \Delta\mathbf{u}(t) - \Delta\mathbf{u}^* \tag{3.3-16}$$

The basic LQR determines the control $\Delta\mathbf{u}(t)$ that minimizes the quadratic cost function

$$J = \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\Delta\tilde{\mathbf{x}}^T \mathbf{Q} \Delta\tilde{\mathbf{x}} + \Delta\tilde{\mathbf{u}}^T \mathbf{R} \Delta\tilde{\mathbf{u}}\right) dt \tag{3.3-17}$$

Equation 3.4-17 is slightly different than the cost function used in defining the OCM (eq. 3.4-1), which contains a control rate weighting. Control dynamics were neglected to simplify controller structure and to minimize the effects of lags in the system. The solution to the LQR problem is a feedback control law of the form

$$\Delta\tilde{\mathbf{u}}(t) = -\mathbf{C}\Delta\tilde{\mathbf{x}}(t) \tag{3.3-18}$$

The matrix $\mathbf{C}$ contains the optimal gains that minimize J, and is found by solving the *Algebraic Riccati equation* associated with the system defined by eq. 3.4-11 and the cost function J. Stability of the closed-loop is guaranteed provided that the system model and the matrices $\mathbf{Q}$ and $\mathbf{R}$ satisfy a number of key conditions (Stengel, 1986). Substituting eqs. 3.4-15 and 3.4-16 into eq. 3.4-18,

$$\Delta\mathbf{u}(t) = \Delta\mathbf{u}^* - \mathbf{C}[\Delta\mathbf{x}(t) - \Delta\mathbf{x}^*] \tag{3.3-19}$$

The variables $\Delta\mathbf{x}^*$ and $\Delta\mathbf{u}^*$ represent some desired steady-state perturbation of state and control variables from a nominal solution. They depend on the desired value of the output $\Delta\mathbf{y}^*$ (defined by eq. 3.4-12). At steady state, $\Delta\dot{\mathbf{x}} = \mathbf{0}$ so that

$$\begin{bmatrix} \mathbf{0} \\ \Delta \mathbf{y}^* \end{bmatrix} = \begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{H_x} & \mathbf{H_u} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}^* \\ \Delta \mathbf{u}^* \end{bmatrix} = \mathbf{A} \begin{bmatrix} \Delta \mathbf{x}^* \\ \Delta \mathbf{u}^* \end{bmatrix} \tag{3.3-20}$$

The dimensions of **x**, **u**, and **y** are n, m, and r, respectively. In the case where $r = m$ (i.e., the system has the same number of commanded outputs as controls), the set point is given by

$$\begin{bmatrix} \Delta \mathbf{x}^* \\ \Delta \mathbf{u}^* \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} \mathbf{0} \\ \Delta \mathbf{y}^* \end{bmatrix} \tag{3.3-21}$$

Define **B** as the inverse of **A**,

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \mathbf{A}^{-1} \tag{3.3-22}$$

where the partitions of **B** have the same dimensions as the partitions of **A** in eq. 3.4-20. The set point is then readily computed as

$$\Delta \mathbf{x}^* = \mathbf{B}_{12} \Delta \mathbf{y}^* \tag{3.3-23}$$

$$\Delta \mathbf{u}^* = \mathbf{B}_{22} \Delta \mathbf{y}^* \tag{3.3-24}$$

where

$$\mathbf{B}_{11} = \mathbf{F}^{-1}(-\mathbf{G}\mathbf{B}_{21} + \mathbf{I}_n) \tag{3.3-25}$$

$$\mathbf{B}_{12} = -\mathbf{F}^{-1}\mathbf{G}\mathbf{B}_{22} \tag{3.3-26}$$

$$\mathbf{B}_{21} = -\mathbf{B}_{22}\mathbf{H}_\mathbf{x}\mathbf{F}^{-1} \tag{3.3-27}$$

$$\mathbf{B}_{22} = (-\mathbf{H}_\mathbf{x}\mathbf{F}^{-1}\mathbf{G} + \mathbf{H}_\mathbf{u})^{-1} \tag{3.3-28}$$

It is apparent that there will be a problem if $(-\mathbf{H}_\mathbf{x}\mathbf{F}^{-1}\mathbf{G} + \mathbf{H}_\mathbf{u})$ is singular. This typically happens if one of the state variables is an integral of one of the elements of **y**. This is a case of *quasistatic equilibrium*, and it can be handled through a modification of the above development.

Substituting eqs. 3.4-23 and 3.4-24 into eq. 3.4-19 yields the non-zero set point regulator in terms of the command variable $\Delta \mathbf{y}^*$:

$$\Delta \mathbf{u}(t) = \mathbf{C}_F \Delta \mathbf{y}^* - \mathbf{C}_B \Delta \mathbf{x}(t) \tag{3.3-29}$$

where

$$\mathbf{C}_F = \mathbf{B}_{22} + \mathbf{C}\mathbf{B}_{12} \tag{3.3-30}$$

$$\mathbf{C}_B = \mathbf{C} \tag{3.3-31}$$

and the matrix **C** is the solution to the LQR problem stated earlier. Thus, given a linear time invariant system (**F**, **G**), control cost function weights (**Q**, **R**) and a desired set of command variables ($\mathbf{H_x}$, $\mathbf{H_u}$), it is possible to design a non-zero set point regulator to optimally track a desired command variable $\Delta\mathbf{y}^*$. The following sections describe the development of a regulator for the longitudinal and lateral/directional dynamics of the UH-1 to provide variable stability augmentation in the IFT.

3.3.2.2 Longitudinal Control Design

For the purposes of the longitudinal control design, the vehicle state and control vectors are defined as

$$\mathbf{x}_{lon} = \begin{bmatrix} u & w & \theta & q & x_1 & x_3 & x_5 \end{bmatrix}^T \tag{3.3-32}$$

$$\mathbf{u}_{lon} = \begin{bmatrix} \delta_{lon} & \delta_{col} \end{bmatrix}^T \tag{3.3-33}$$

Altitude is not included in $\mathbf{x}_{lon}$ to prevent the problem of quasistatic equilibrium mentioned in the previous section. The command variables for the regulator are the control settings themselves, i.e., $\mathbf{y}_{lon} = \mathbf{u}_{lon}$.

The **Q** and **R** matrices for the LQR cost function are defined using the "inverse square rule" (Bryson & Ho, 1975); i.e., they are diagonal matrices whose elements are interpreted as the reciprocal of the allowable mean-squared values of state and control perturbations. Thus,

$$Q_{ii} = 1 / \Delta x_{i_{\max}}{}^2 \tag{3.3-34}$$

$$R_{ii} = 1 / \Delta u_{i_{\max}}{}^2 \tag{3.3-35}$$

The cost function uses state weighting on altitude rate $\Delta\dot{z}$ and forward speed Δu. Although altitude is not an element of $\mathbf{x}_{lon}$, its rate of change may be expressed as

$$\Delta\dot{z} = \mathbf{h}^T \Delta\mathbf{x}_{lon} \tag{3.3-36}$$

where the elements of **h** are determined from the original motion equations. Given an allowable mean-squared altitude rate variation $\Delta\dot{z}_{\max}$, eq. 3.4-36 can be used to determine the equivalent **Q** components. Table 3.3-11 lists the mean-squared perturbation values that define the control gains at the highest augmentation level (i.e., the highest gains) at all flight speeds.

Table 3.3-11: Longitudinal Cost Function Elements

Variable	Value
$\Delta u_{\max}$	2.0 ft/sec
$\Delta \dot{z}_{\max}$	2.0 ft/sec
$\Delta \delta_{lon_{\max}}$	0.3 inches
$\Delta \delta_{col_{\max}}$	0.3 inches

Given matrices $\mathbf{Q}_{\max}$ and $\mathbf{R}_{\max}$ that define the highest augmentation level $k_{\max}$ (using Table 3.3-1 and eqs. 3.4-34 - 3.4-36), the question arises of how to define $\mathbf{Q}$ and $\mathbf{R}$ for all lower augmentation levels. As discussed earlier, the intent of the helper is to augment the stability of the vehicle and make it easier for the student to control at higher "help levels." As the help level decreases, the amount of augmentation (i.e., the control gains) should decrease so that at the lowest level, the student is flying an unaugmented helicopter (so that the feedback gain matrix $\mathbf{C} \approx \mathbf{0}$). This can be accomplished by defining $\mathbf{Q}_k$ and $\mathbf{R}_k$ for the k^{th} augmentation level as follows:

$$\mathbf{Q}_k = \rho(k)\mathbf{Q}_{\max} \quad k = 0,\ldots,k_{\max} \tag{3.3-37}$$

$$\mathbf{R}_k = \mathbf{R}_{\max} \tag{3.3-38}$$

where

$$\rho(k) = 1.4^{k-k_{\max}} \tag{3.3-39}$$

Figure 3.3-4 illustrates $\rho(k)$ for $k_{\max} = 19$. At $k = k_{\max}$, $\rho = 1$ so that $\mathbf{Q} = \mathbf{Q}_{\max}$. As $k \rightarrow 0$, $\rho \rightarrow 0$ so that $\mathbf{Q} \rightarrow 0$. Provided that the open-loop system is stable, the LQR feedback gains will tend to zero as the elements of $\mathbf{Q}$ diminish.* This formulation of the LQR problem produced subjectively acceptable changes in closed-loop stability with decreasing help level. Although a linear variation of $\mathbf{Q}$ with help level might seem the most straightforward approach, it was found that this resulted in very large subjective changes in vehicle stability at the lowest help levels that were difficult for students to handle. By making the changes in cost function weights more gradual at lower help levels, students were able to make the transition to flying an unaugmented helicopter more easily.

* If the open-loop system is unstable, the control gains will not vanish and the closed-loop system achieves a "minimum control energy" configuration.

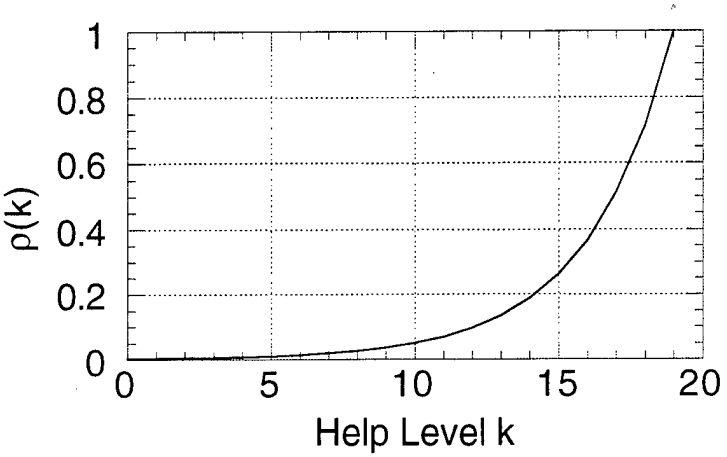


Figure 3.3-4: Help Level Scaling Function for Definition of Weighting Matrix Q

Using these definitions, feedforward and feedback matrices $\mathbf{C}_F$ and $\mathbf{C}_B$ (eqs. 3.4-30 and 3.4-31) were computed for all 20 help levels (0 to 19) and 9 forward flight speeds between 0 ft/sec and 160 ft/sec, in steps of 20 ft/sec. During simulated flight, the gains were linearly interpolated as a function of airspeed.

Figure 3.3-5 illustrates the step response characteristics of the system. Shown is the u response to a step input (of magnitude 0.2) in longitudinal cyclic from a hover condition for three help levels: 0 (i.e., the open-loop response), 14, and 19 (the maximum help level). The open loop response is quite sluggish and has a very long settling time. The highest help level produces a rapid, highly damped response. As desired, the intermediate help level produces a response in between the two extremes.

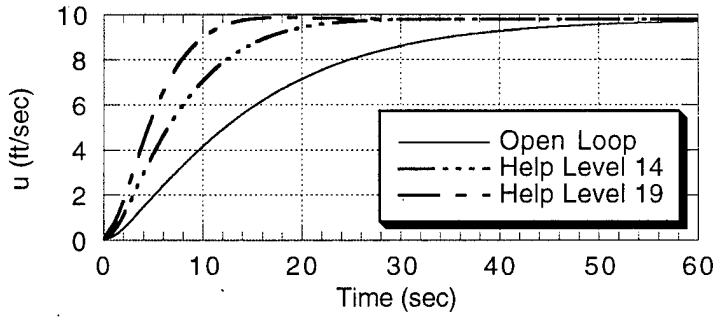


Figure 3.3-5: Step Response in u to a longitudinal cyclic input

3.3.2.3 Lateral-Directional Control Design

The state and control for lateral-directional control law are defined as

$$\mathbf{x}_{lat} = \begin{bmatrix} v & p & r & \phi & x_2 & x_4 \end{bmatrix}^T \tag{3.3-40}$$

$$\mathbf{u}_{lat} = \begin{bmatrix} \delta_{lat} & \delta_{ped} \end{bmatrix}^T \tag{3.3-41}$$

The formulation of the lateral control law is identical to that of the longitudinal controller. The lateral cost function uses weighting on roll and yaw rates Δp and Δr, as well as weighting on both control inputs. Table 3.3-12 lists the allowable mean-squared variations that define the highest augmentation level. Cost function weights are scaled as a function of augmentation level in the same manner as the longitudinal regulator.

The command variables for the lateral/directional regulator are defined as

$$\mathbf{y}_{lat} = \begin{bmatrix} \delta_{lat} & \delta_{ped} \end{bmatrix}^T \tag{3.3-42}$$

During simulation, perturbation states $\Delta\mathbf{x}_{lon}$, $\Delta\mathbf{x}_{lat}$ and perturbation controls $\Delta\mathbf{u}_{lon}$, $\Delta\mathbf{u}_{lat}$ are computed using eqs. 3.4-6 and 3.4-7 (with appropriate subscripts inserted into these equations). The nominal state and control are computed via a two-dimensional lookup based on altitude and airspeed. Altitude dependence is included in defining the nominal state and control because of the ground effect model, which affects the equilibrium conditions at low altitudes. The perturbation command variables $\Delta\mathbf{y}_{lon}$, $\Delta\mathbf{y}_{lat}$ are defined as

$$\begin{aligned} \Delta\mathbf{y}_{lon}(t) &= \mathbf{y}_{lon}(t) - \mathbf{y}_{o_{lon}}(t) \\ &= \mathbf{u}_{lon}(t) - \mathbf{u}_{o_{lon}}(t) \end{aligned} \tag{3.3-43}$$

$$\begin{aligned} \Delta\mathbf{y}_{lat}(t) &= \mathbf{y}_{lat}(t) - \mathbf{y}_{o_{lat}}(t) \\ &= \mathbf{u}_{lat}(t) - \mathbf{u}_{o_{lat}}(t) \end{aligned} \tag{3.3-44}$$

where $\mathbf{u}_{lon}$ and $\mathbf{u}_{lat}$ are the absolute control positions as commanded by the pilot. Equation 3.4-29 then defines the feedback control input for both the longitudinal and lateral regulators.

Table 3.3-12: Lateral Cost Function Elements

Variable	Value
$\Delta p_{\max}$	5.0 deg/sec
$\Delta r_{\max}$	3.0 deg/sec
$\Delta\delta_{lat_{\max}}$	0.9 inches
$\Delta\delta_{ped_{\max}}$	0.9 inches

3.4 Hardware Implementation

In the course of this Phase II effort, the Training Research Simulator (TRS) at the Army Research Institute's (ARI) Aviation R&D Facility has transitioned from the MicroVAX-based architecture described by Zacharias *et al* (1993) to a personal computer (PC) based configuration. Figure 3.4-1 illustrates the architecture currently installed at Ft. Rucker. A 486-class PC (labeled the SIM) hosts the equations of motion of a UH-1 helicopter. This PC communicates with the cockpit interface processor (CIP) and the IFT via an Ethernet connection. The cockpit interface processor (CIP) provides the interface between the pilot in the OH-58 helicopter cab and the vehicle model hosted on the SIM computer. The CIP hosts a data acquisition system that performs A/D conversion on the position and settings of cockpit controls and switches. On each simulation timestep, the SIM receives pilot control inputs sampled by the CIP and helper inputs from the IFT, which together drive the motion equations. The SIM sends instrument readings back to the CIP, which are passed on to the SGI display host for presentation to the pilot on an electronic flight instrumentation system (EFIS) that duplicates the primary flight instruments in a TH-67 helicopter. The original system design had called on the CIP to display the flight instruments, but we found that LabView could not draw the instruments quickly enough on the 486 platform to maintain a 1/30 sec loop time. Although currently disabled, the CIP code does contain modules to display the instruments directly, should future computer hardware modifications render the CIP fast enough to display the instruments with an adequate refresh rate. The CIP is also responsible for coordinating the trim logic, which assists the student in prepositioning the controls for a stationary hover or other start-up flight maneuver.

The SIM also sends the aircraft's state components to the IFT, which uses them to compute helper inputs and to drive the expert system shell providing voice feedback. The speech board currently in use is unable to operate asynchronously; as such, it was necessary to place it on a separate PC and transmit the text of IFT verbal messages from the IFT to that PC across a serial line. The out-the-window display could be generated by low-cost PC-hosted image generators now produced by a variety of vendors, but to minimize development time and costs, two available BBN 120 TX image generators were used. They communicate directly with the SIM through dedicated interface hardware.

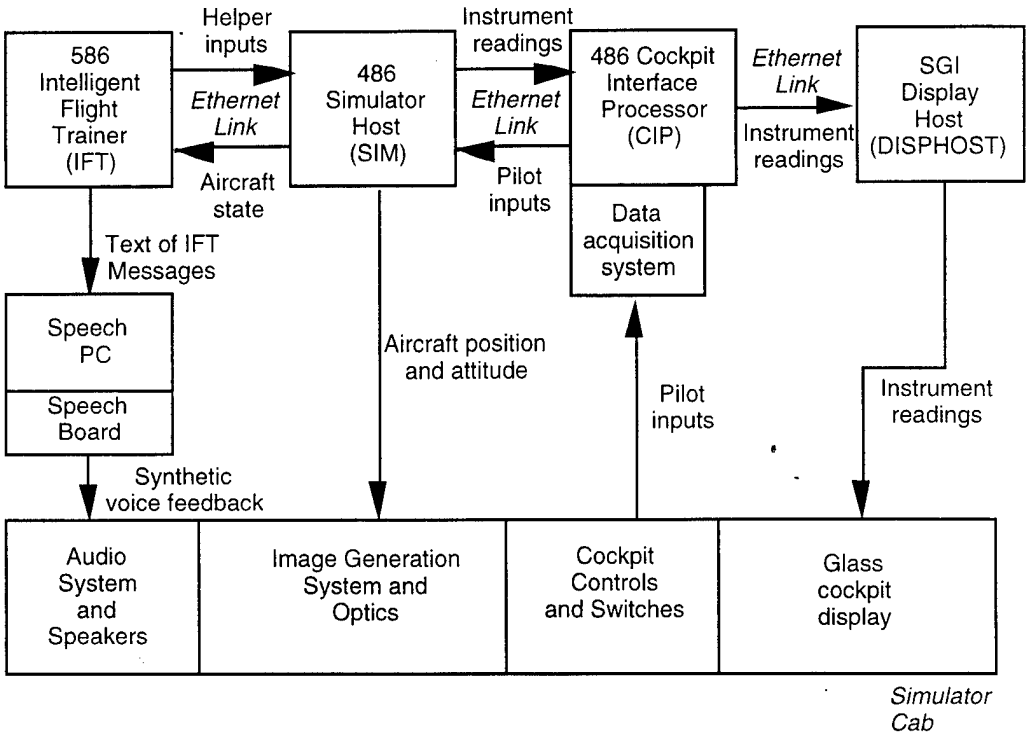


Figure 3.4-1: IFT Functional Block Diagram

The distributed computing environment described here reduces the computational load on the SIM by isolating all IFT functions on a separate PC. As such, the IFT and the SIM effectively function as parallel processors. Figure 3.4-2 illustrates the system's overall timing diagram. We selected this arrangement of computational operations and communication to minimize the amount of time that any computer spent waiting for data packets to arrive from another machine, to make the most of available computing time on each loop. In the figure, the index i denotes quantities associated with the i^{th} iteration of the loop. At the beginning of the i^{th} loop, the SIM transmits the instrument data (stored in the vector $\mathbf{y}$) computed in the previous iteration to the CIP. The SIM then transmits the vehicle state vector $\mathbf{x}$ and the vector of pilot controls $\mathbf{u}$ to the IFT. The SIM updates the vehicle equations of motion using the current state and the pilot control inputs from the previous loop. While this is going on, the IFT computes its helper control inputs $\mathbf{u}_{aug}$ and the CIP samples and scales the cockpit controls. After updating the vehicle state, the SIM sends a set of transformation matrices describing the current position and attitude to the image generators, which uses them to generate the out-of-window display. Then SIM then receives the pilot controls transmitted from the CIP, and the augmenting control inputs from the IFT. Both of these quantities are used together to update the vehicle state on the next iteration. After transmitting the augmenting control inputs to the SIM, the IFT advisor examines the vehicle state components and produces verbal feedback for the student, as necessary. After

sending the pilot control inputs to the SIM, the CIP updates the instrument display using the data it received earlier from the SIM. Although not shown, this process entails the transmission of the instrument data from the CIP to the SGI workstation in the loop, which actually provides the glass cockpit display. Finally, the CIP computes and provides a variable time delay to fix the overall loop time to 33 msec, to provide a 30 Hz update rate.

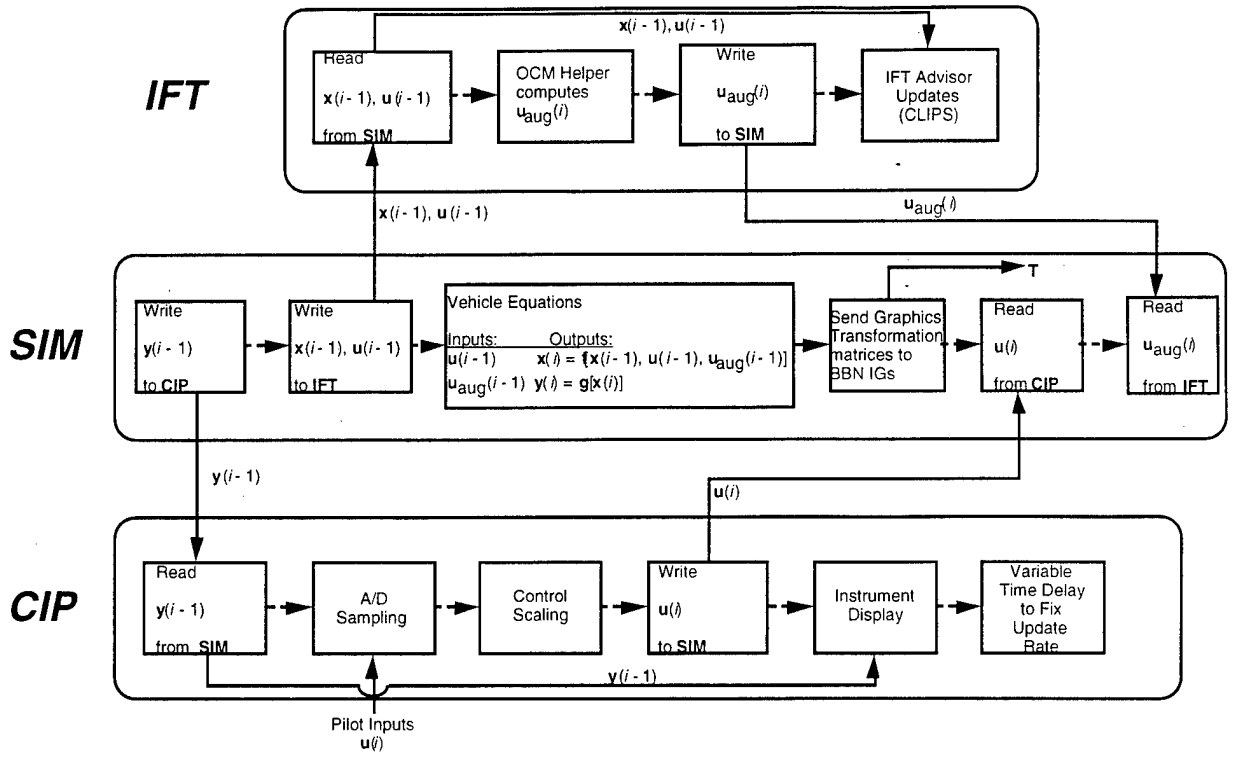


Figure 3.4-2: System Timing Diagram

The complete operation of the CIP and its coordination with the SIM and IFT are described in Appendices A and B. Appendix C describes the CIP/cockpit wiring interface. Appendix D describes a set of analog-digital (A/D) calibration routines, and Appendix E describes the Ethernet communication protocol between the CIP and the SGI display host. Finally, Appendix F contains a number of photographs of the installed setup at Fort Rucker.

4. System Evaluation

This chapter summarizes results of the system demonstration and validation effort. Section 4.1 provides a brief summary of model validation training transfer study results using the UH-1 TRS. Section 4.2 presents training results for the maneuver sets developed under Phase II of this effort.

4.1 UH-1TRS Validation

Much of the early work on validation has focused on the effectiveness of the UH-1TRS as an IERW trainer. The UH-1TRS has been used in six separate research studies by Dohme and colleagues at ARIARDA at Ft. Rucker, and it has undergone considerable modification over the course of these studies. To put the current effort in historical perspective, we very briefly review these studies in the following paragraphs. A more complete review can be found in Dohme & Longridge (1988) and Dohme (1992).

Study 1: An initial study was conducted in June 1988, to evaluate the potential of the newly constructed UH-1TRS. At that time, the trainer incorporated a relatively primitive linearized UH-1 model (Bailey & Krishnakumar, 1986) derived from a rotor disk model derived by Talbot & Corliss (1977), informally known as the NASA ARC Uncle model. The visuals were implemented with a two-channel IRIS 2400T configuration (in contrast to the current BBN configuration) which supported a relatively low frame rate of about 15 Hz, a low polygon count of 300, no terrain undulations, and no texturing. In spite of these limitations, a structured transfer-of-training (TOT) experiment showed a moderate positive transfer effectiveness ratio (TER) (Roscoe & Williges, 1980), for eight Primary Phase maneuvers, including hover, taxi, turn, auto, takeoff, pattern, approach, and land.

Study 2: A second study in early spring 1989 was conducted to evaluate the effectiveness of system improvements in the aerodynamic model and the visuals. The helicopter model was upgraded by developing a ground effect model, and by tweaking the aerodynamic parameters to improve the handling qualities. The visuals were upgraded by going to the BBN 120TX/T's which raised the frame rate to 30 Hz, raised the polygon count to 1000, supported a realistic terrain model of Ft. Knox, and supported texturing needed for low-speed visual flow. A second TOT experiment showed a moderate improvement in TER's compared with those found in the first study. The source of this may have been the improved fidelity of the second configuration, but it may also have been the result of using neophyte trainees, whose normally rapid initial skill acquisition might be expected to contribute to very large TER's, no matter what the simulator configuration.

Study 3: A third study in the fall of 1989 demonstrated the effectiveness of yet greater fidelity in the visuals. Here the visuals were upgraded by going to Evans and Sutherland (E&S) ESIG 500H's, which raised the frame rate to 50 Hz, improved the level-of-detail and texture management functions, and supported additional features such as weather. A corresponding TOT experiment showed yet more improvement in the TER's, when compared with those in the second study, with the overall average higher and with all individual TER's distinctly positive.

Study 4: A fourth study showed how the UH-1TRS could be used in a substitution mode rather than an adjunct to training, as it had been in the previous three studies. Here time on the simulator replaced actual blade hour training in the aircraft, on a one-for-one basis. The simulator configuration was virtually identical to that used in Study 2, with the mid-range BBN 120TX/T's used as the CIG's. Results showed that positive TOT was achieved in six of the eight trained maneuvers, with an overall positive TER. A significant savings in flight time hours, and thus dollars, was achieved.

Study 5: A fifth study focused on one of the eight previously-studied maneuvers, namely hover, and evaluated the contribution of the motion base, in a classic motion/no-motion paradigm. The configuration was essentially the same as that used in the previous study. The results showed that motion during training did not significantly affect either training rate to criterion level on the simulator, or checkride score in the aircraft. The results confirmed a number of earlier studies on motion training effectiveness.

Study 6: The UH-1TRS was upgraded to incorporate an AHT autohelper, which helps the student pilot control the vehicle, via stability augmentation of the helicopter math model (Dohme, 1990; Dohme, 1991). Autohelp level is chosen as a function of student performance, starting at high levels of augmentation with a naive and thrashing student , and progressing to a zero level of augmentation as the student becomes more proficient. The AHT performance monitoring, augmentation, and switching software was hosted on the FPS AP, and this is the current configuration which now exists. A training evaluation experiment focused on hover, and showed that after three hours on the AHT, student pilots could attain a passing score of between 85% to 100% on an in-flight evaluation test usually administered after 12 to 14 hours of instructor pilot flight training.

Recent Developments: In the past year, the TRS architecture has transitioned from the Vax-based architecture described in the Phase I report of this study to the PC-based architecture described earlier in this report. The hardware has been physically relocated to another building at Ft. Rucker, and a new simulator cab has been built from a discarded OH-58 cockpit. Recent efforts in the area of model verification have focused on tuning the basic vehicle dynamics,

adding engine and ETL models, tuning the ground effect representation, and verifying the subjective handling qualities produced by the gain-scheduled helper. With guidance from retired Army pilots who served as test subjects, some ad-hoc modifications were made to the vehicle dynamics and the helper to improve their control response. Specifically, the control effectiveness of the bare airframe was increased at low airspeeds, in response to comments that the vehicle was unresponsive to pilot inputs at hover. Additionally, gain-scheduled helper outputs were scaled *down* in ground effect, after it was discovered that the helper (which was designed for out-of-ground-effect flight) produced a very "twitchy" response near the ground.

4.2 Training Results

Following Phase II system design enhancement, implementation, testing, and tuning, we conducted a brief demonstration and evaluation of student pilot training with the IFT. Our findings here focus primarily on the helper and advisor modules. A neophyte student trainee with a basic understanding of flight principles but minimal flight experience was trained on hover, hover taxi, hover turn, and traffic pattern maneuvers. The student was instructed to follow the IFT's voice feedback and was given no additional verbal cueing by the instructor during simulation.

4.2.1 Fixed-Point Hover

Table 4.2-1 lists the allowable error thresholds for the hover maneuver. Figure 4.2-1 presents the student's x, y, and h errors vs. time during the hover maneuver. Figure 4.2-2 shows the ψ errors vs. time. Finally, Fig. 4.2-3 shows the corresponding help level. Training was initiated at a help level of 10. Table 4.2-2 lists the sequence of verbal messages that the IFT issued to the student during the training sequence.

Table 4.2-1: Hover Maneuver Error Thresholds

Variable	Threshold
x	± 25 ft
y	± 5 ft
h	± 3 ft
ψ	± 10°

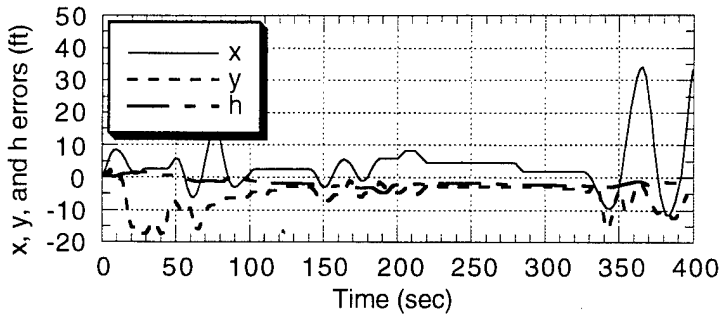


Figure 4.2-1: Position Errors vs. Time during Hover

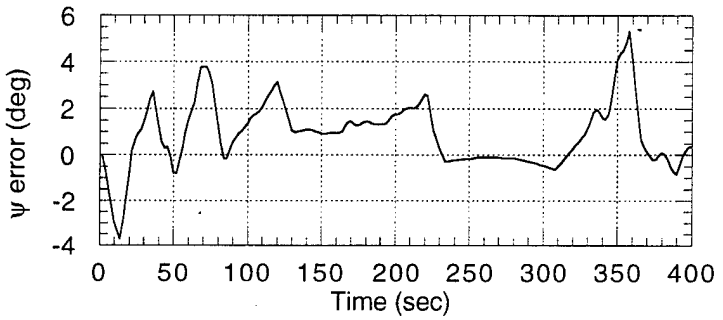


Figure 4.2-2: Heading Errors vs. Time during Hover

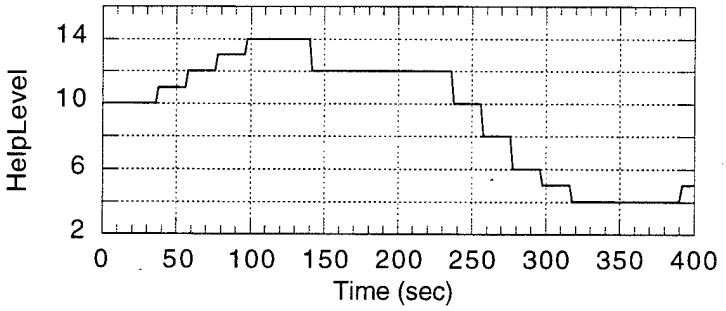


Figure 4.2-3: Help Level vs. Time during Hover

Figure 4.2-2 indicates that the student kept ψ within the acceptable bounds (given in Table 4.2-1) throughout the entire maneuver. At first, the student had considerable difficulty maintaining y within bounds, which led to the help level increasing to 14 by t = 100 sec. Help level then steadily decreased to 4, indicating that the student was slowly learning to maintain the fixed hover with decreasing augmentation. Near the end of the training session, the student began a large amplitude oscillation in x, which led to an increase in help level just before the end of the maneuver. The instructor then terminated the session to allow the student to re-focus his attention before trying the maneuver again.

Table 4.2-2: IFT Messages during Hover

Time (sec)	Message
16	Check your location over the runway.
26	You are too far left.
36	You are too far left. Help level is increasing.
46	You are too far left; move right.
56	You are too far left. Help level is increasing.
66	Move right using right cyclic.
76	You are too far left. Help level is increasing.
86	Check your location over the runway.
96	You are too far left. Help level is increasing.
106	You are too far left.
140	Help level is decreasing.
150	You are too far left; move right.
175	Move right using right cyclic.
200	Check altitude.
236	Help level is decreasing.
256	Help level is decreasing.
276	Help level is decreasing.
296	Help level is decreasing.
316	Help level is decreasing.
341	Check your location over the runway.
351	You are too far left.
361	Check your location over the runway.
371	You are too far forward.
381	You are too far left; move right.
391	You are too far left. Help level is increasing.

4.2.2 Hover Turn

Figures 4.2-4 to 4.2-6 illustrate the same student's performance during a hover turn. Table 4.2-3 lists the corresponding error thresholds. The nominal desired turn rate was 3°/sec. Figure 4.2-4 shows the x, y, and h errors, while Fig. 4.2-5 shows the helicopter's absolute heading angle and turn rate. Finally, the IFT help level is shown in Fig. 4.2-6.

Table 4.2-3: Error Thresholds for Hover Turn

Variable	Threshold
x	± 25 ft
y	± 25 ft
h	± 3 ft
$\dot{\psi}$	± 1.5 °/sec

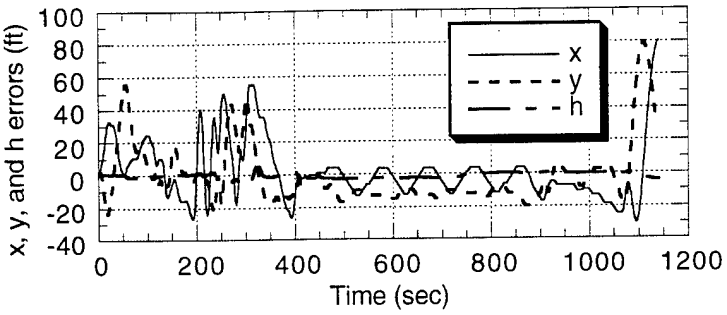


Figure 4.2-4: Position Errors vs. Time during Hover Turn

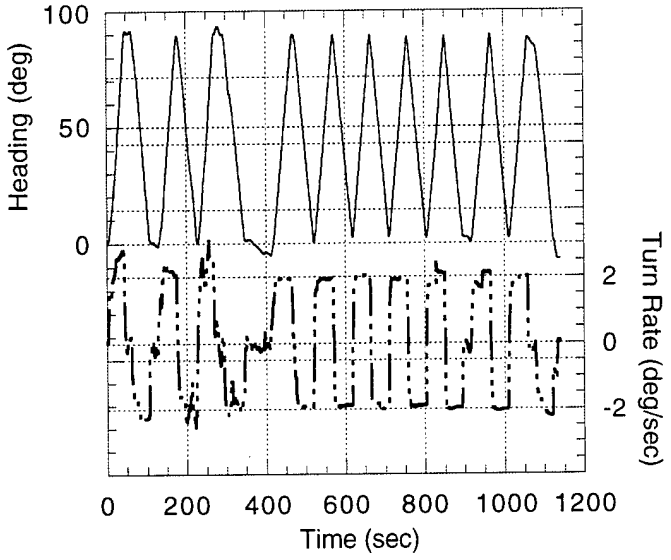


Figure 4.2-5: Heading and Turn Rate vs. Time during a Hover Turn

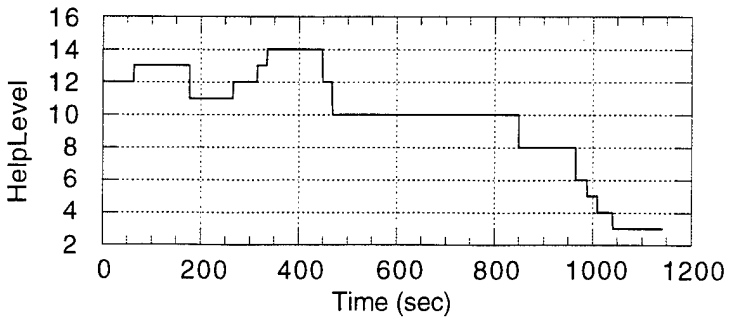


Figure 4.2-6: Help Level vs. Time during Hover Turn

Initially, the student had considerable difficulty in maintaining the nominal pivot point and in making smooth alternating turns. The help level increased to 14 by $t = 350$ sec, at which point the student regained control over the helicopter's drift and established smooth alternating turns near the nominal pivot point. Over the following 600 sec (10 min), the student was able to reduce the help level down to 2. However, once the help level reached 2 the student could not maintain position and drifted far away from the pivot point. At this point, the maneuver had lasted over 18 minutes and the instructor decided to allow the student to rest before trying the maneuver again.

4.2.3 Traffic Pattern

The final evaluation of the IFT was performed with the traffic pattern maneuver, which incorporates the gain-scheduled helper described in Section 3.4.2. Table 4.2-4 lists the error thresholds used during a traffic pattern. The nominal flight parameters were defined as in Section 3.4.1.4. Figures 4.2-7 - 4.2-9 show the student's performance in his *first* attempt to perform this maneuver. Figure 4.2-7 plots x against y, and shows that the student did a reasonable job in matching the overall shape of the traffic pattern (as illustrated in Fig. 3.4-2). Figure 4.2-8 shows the airspeed and altitude profiles during the maneuver. The solid lines represent the student's actual velocity and altitude histories, while the dotted lines represent the nominal flight parameters. Finally, Fig. 4.2-9 shows IFT help level vs. time.

The student was unable to effect a net reduction in help level in the course of the traffic pattern. Since this maneuver is relatively complex, it is reasonable to expect that students will have difficulty matching the desired flight parameters on early attempts. However, the results are encouraging because they demonstrate that the neophyte student can follow the *procedural* guidance given verbally by the IFT on when to initiate turns, climbs, descents, etc., and coarsely follow the desired trajectory. A key issue will be to examine a student's long term improvement in traffic pattern proficiency over several attempts.

Table 4.2-4: Traffic Pattern Error Thresholds

Variable	Threshold
V	± 10 kt
y	± 100 ft
h	± 100 ft
$\dot{h}$	± 200 ft/min
ψ	± 10°
$\dot{\psi}$	± 1.5 deg/sec

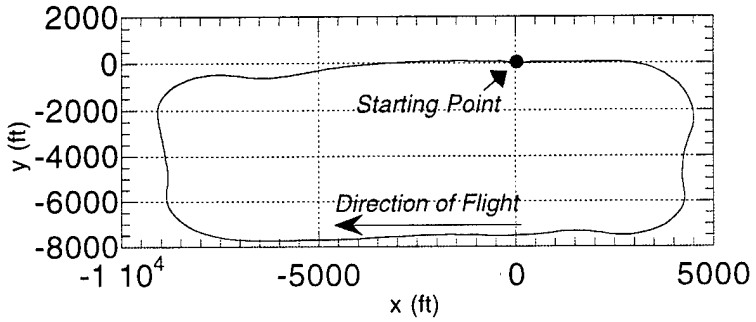


Figure 4.2-7: y vs. x during a Traffic Pattern

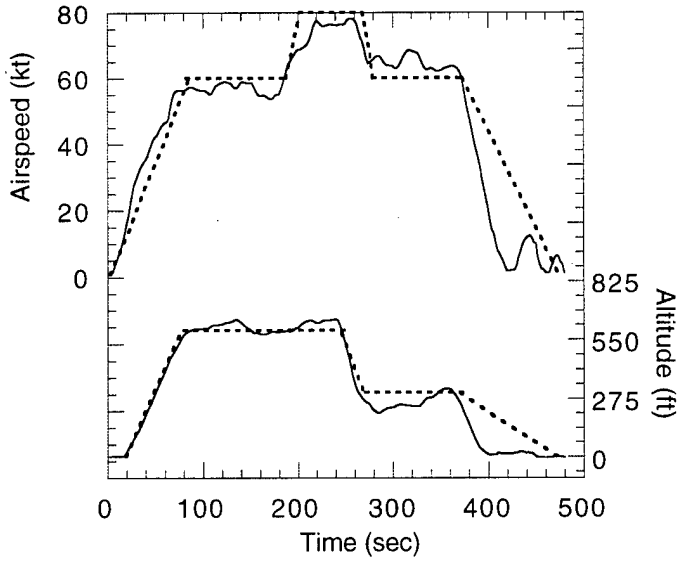


Figure 4.2-8: Airspeed and Altitude vs. Time during Traffic Pattern

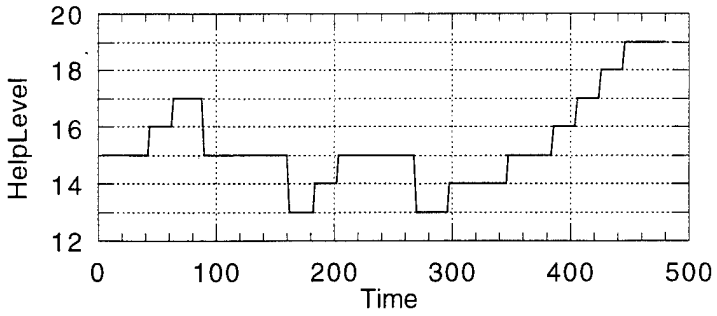


Figure 4.2-9: IFT Help Level vs. Time during Traffic Pattern

4.3 Summary of Demonstration Results

The simulation results presented here show that the IFT is capable of assisting students in developing proficiency at initial entry rotary wing maneuvers. In the hover and hover turn maneuvers, the student reduced the IFT's help level through sustained satisfactory performance. In both cases, performance deteriorated at the lowest help levels, most likely as a result of a drop in attentional focus, fatigue, or changes in vehicle stability that were difficult to handle. The latter might be alleviated by increasing the time interval at each help level, to give the student more time to adjust to the changes in vehicle dynamics. Further tuning of the helper may be necessary to ease the transition to the unaugmented airframe (i.e., zero help).

The traffic pattern demonstration produced mixed results, but on the whole it is very encouraging. On the early attempt at performing this maneuver documented here, the student was unable to effect a net reduction in IFT help level between the beginning and the end of the traffic pattern maneuver. However, to an extent this may be expected, given the complicated nature of this maneuver and the extensive procedural guidance given by the IFT, which far exceeds that in the hover, hover turn, or hover taxi maneuvers. Since the student was able to properly follow the procedural guidance given by the IFT and bring the helicopter to its final resting position after flying a maneuver that generally approximated the ideal traffic pattern, we believe that the IFT clearly has the potential to teach a student how to perform this maneuver. If a student is given repeated opportunities to conduct this maneuver in the IFT, we believe that the student will be able to improve his/her manual tracking performance over time.

5. Summary, Conclusions, And Recommendations

5.1 Summary of Effort

The Phase II project documented in this report focused on the development of a full-scope Intelligent Flight Trainer for initial entry rotary-wing training. We now summarize the major efforts and findings of this project.

We first **designed, developed, and implemented a PC-based system architecture** to exploit the advantages of parallel processing and high-speed Ethernet communication. The initial design for the PC-based architecture employed three computers: 1) the simulator model host; 2) the IFT; and 3) the cockpit interface processor (CIP), intended to provide the visual interface between the student in the vehicle cab and the simulator model/IFT. The simulator host contains the UH-1 equations of motion, which are updated at each time step using pilot inputs provided to the computer by the CIP, which samples pilot inputs at a 30 Hz rate. On each time step, the simulator host transmits the vehicle state and control position information to the IFT, which uses them to assess the student's performance at the intended maneuver. This assessment provides the basis for the IFT advisor's synthetic voice feedback (using text-to-speech software) to the student via a speech board. It was originally intended that the speech board would be hosted within the IFT machine. However, we discovered that the installed speech board was incapable of asynchronous operation (i.e., the computer could not "talk" to the student while performing other computational functions). In the interest of minimizing development time and costs, we chose to continue to use the same speech board installed on a fourth PC, which communicates with the IFT using an asynchronous serial connection. The IFT also computes feedback control inputs for stability augmentation, which are sent back to the simulator host for input into the vehicle dynamics. After updating the vehicle state, the simulator host transmits instrument readings to the CIP, which displays them to the student on a glass cockpit display. Originally, our design had called upon using the LabView environment to perform all control sampling and instrument display functions. However, we discovered that LabView was not capable of maintaining a 30 Hz update rate with all of the necessary instruments. This necessitated the addition of a fifth computer to the system (a Silicon Graphics workstation) for actually drawing the cockpit instruments. The simulator host also transmits vehicle position and attitude data to the BBN 120-TX image generators, which produces the out-of-window display that the student sees in the cockpit cab. In our Phase II effort, the simulator host PC was made to communicate directly with the image generators using dedicated interface hardware. We demonstrated complete end-to-end system performance at an update rate of 30 Hz.

Following successful initial implementation of the PC-hosted system, we focused our attention on **expanding the scope of the IFT's training modules**. In the Phase I effort the IFT's functionality was limited to training a fixed-point hover maneuver. In Phase II we expanded the IFT's scope to include hover taxi, hover turn, traffic pattern, land from hover, and takeoff to hover maneuvers. For each of these maneuvers, we reviewed Army training guidelines (U.S. Army, 1994) and conducted knowledge engineering exercises with the Contracting Officer's Technical Representative and others at Fort Rucker to identify suitable strategies and message sets for training these maneuvers. These exercises provided the basis for developing CLIPS (Giarratano, 1993) expert system modules for training each of the maneuvers. We also extended the functionality of the helper control logic so that it could provide effective stability augmentation across forward flight speeds for a range of "help levels."

Upon completion of the substantive engineering development, we **demonstrated the system's operation** by conducting a set of exercises to evaluate the IFT's ability to train a neophyte student in the basic flight maneuvers. The student demonstrated sustained improved performance in the hover and hover turn maneuvers, and properly followed the IFT's procedural guidance during a traffic pattern. The results are encouraging, and they set the stage for a more formal evaluation of the IFT's training potential in a transfer of training experiment.

5.2 Conclusions

The results of this Phase II effort demonstrate the feasibility of simulator-based preliminary helicopter flight training with an intelligent tutoring system concept. We have extended the hover trainer developed in the predecessor Phase I effort into a device of much wider scope that can train a set of introductory flight maneuvers. The successful PC-based implementation demonstrates that low-cost computing technologies can deliver a system with the needed capacity.

The demonstration results presented in Chapter 4 show that the combination of synthetic voice feedback and adaptive stability augmentation can help a student improve his/her basic flight proficiency in the simulated environment. Our Phase II effort extended the IFT from a system that only provided feedback on manual task performance during a fixed-point hover into one that provides important procedural cueing on the commencement and termination of various flight segments such as accelerations, decelerations, level flight, climbs, descents, turns, final approach, landing, and takeoff. The traffic pattern training results, while preliminary, show that a student can comprehend and follow these instructions to conduct a relatively complicated flight task without any additional instructions from a human instructor pilot. We believe that these results demonstrate the potential of simulator-based flight training with intelligent tutoring, and

that with a continued development effort a fully-functional production training system can be designed and implemented. Our results suggest that such a system offers considerable potential to reduce dependence on an instructor pilot during preliminary flight training.

5.3 Recommendations for Follow-On Development

In the course of conducting this Phase II design, development, and demonstration effort, we have developed numerous insights pertaining to the development of intelligent flight training devices. We now summarize our major recommendations for transitioning the research prototype developed here into a fully functional training tool:

1) **Conduct a formal validation and transfer of training experiment**: The IFT's ability to teach students how to perform all of its maneuvers must be fully tested for a meaningful sample of students. The IFT's ability to deliver positive *transfer of training* to the flight line must also be determined.

2) **Transition to the TH-67 helicopter:** The current implementation and its VAX-based predecessor at the ARI Simulator Facility at Fort Rucker use a UH-1H vehicle model. Since current Army flight training curricula specify the use of the TH-67 helicopter, it is appropriate to replace the UH-1H simulation model with one of the TH-67 that provides sufficient fidelity to meet preliminary flight training requirements.

3) **Develop training modules for conducting "air work"**: The IFT presently can train the following maneuvers: hover, hover taxi, hover turn, traffic pattern, takeoff to hover, and land from hover. These maneuvers were selected through consultation with the Contracting Officer's Technical Representative. There is a considerable increase in complexity and difficulty between the so-called "hover maneuvers" and the traffic pattern. In order to help students learn how to handle the helicopter during high-speed forward flight, it would be appropriate to have them practice short finite-duration maneuvers such as level flight, climbs, turns, descents, etc., before conducting a full-fledged traffic pattern. This would help students develop manual control proficiency before attempting the challenging traffic pattern.

4) **Install all system components on a single PC**: In the three years since this Phase II effort was awarded, there has been a considerable improvement in the computing power of personal computers. Modern high-speed PCs based on the Pentium chip (and its soon-to-come successor, the P6) provide enough raw computing power to perform all IFT/SIM computational tasks at a sufficient speed. In the past year, numerous vendors have introduced low-cost textured image generator boards for PCs that can deliver multi-channel textured workstation-quality graphics at a fraction of the cost. Consolidation of all

functions on a single machine would drastically simplify system design and operation, and provide greater flexibility in specification of graphical content on the instrument panel and the out-of-window display. For example, "highway-in-the-sky" displays might be used to assist students in tracking a nominal approach path. Or, logic could be developed to highlight specific instruments on the glass cockpit display when the related variable is out of tolerance, to help the student focus his/her attention on that quantity. This could provide for a kind of "adaptive display augmentation," to complement the stability augmentation already provided by the IFT. Such a system might provide additional visual cues in the out-of-window display and on the instrument panel at high help levels, and gradually remove these elements as the student demonstrates performance proficiency improvement.

6. References

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). "Intelligent Tutoring Systems". *Science*, 228, 456-467.

Bailey, J. E., & Krishnakumar, K. (1986). *VICOPS UH-1 Flight Dynamics Model and Software Development* (Report BER 378-29): Univ. of Alabama College of Engineering.

Bailey, J. E., Krishnakumar, K., Whorton, M. S., & Suman, S. K. (1988). *UH-1 Training Research Simulator Flight Dynamics Model Development and Evluation* (Final Report BER 437-177): Univ. of Alabama College of Engineering (May).

Brown, J. S., Burton, R. R., & deKleer, J. (1982). "Pedagogical, Natural Language, and Knowledge Engineering Techniques in SOPHIE I, II, and III". In D. Sleeman & J. Brown (Eds.), *Intelligent Tutoring Systems*, . London: Academic Press.

Bryson, A. E., & Ho, Y. C. (1975). *Applied Optimal Control.* Washington, D.C.: Hemisphere Publishing.

Burton, R. R. (1982). "Diagnosing Bugs in a Simple Procedural Skill". In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, . London: Academic Press.

Dohme, D. A., & Longridge, T. M. (1988). *Transfer of Training from a Low Cost Helicopter Simulator to the UH-1 Aircraft* (ARI Working Paper): Fort Rucker, AL: Army Research Institute Aviation Research and Development Activity.

Dohme, J. (1990). *Low Cost Helicopter Simulation for Primary Flight Training* (ARIARDA Technical Memo): Army Research Institute Aviation R & D Activity

Dohme, J. (1991). "Transfer of Training from a Low Cost Helicopter Simulator". *Symposium on Aviation Psychology*, Ohio State University.

Dohme, J. A. (1992). *Automated Hover Training for Army Flight Students* : Fort Rucker, AL: Army Research Institute Aviation Research and Development Activity

Gai, E. G., & Curry, R. E. (1976). "A Model of the Human Observer in Failure Detection Tasks". *IEEE Trans. on Systems, Man and Cybernetics*, 6.

Giarratano, J., (1993). *CLIPS User's Guide Version 6.0.* NASA Lyndon B. Johnson Space Center, Software Technology Branch.

Goldstein, I. P. (1982). "The Genetic Graph: A Representation for the Evolution of Procedural Knowledge". In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, . London: Academic Press.

Gonsalves, P. G., Kneller, E. W., & Zacharias, G. L. (1989b). *Model-Based Method for Terrain-Following Display Design* (AAMRL-TR-89-039): Charles River Analytics Inc. (June).

Gutstein, E. (1992). "Using Expert Computer Knowledge to Design a Self-Improving Intelligent Tutoring System". In C. Frasson (Ed.), *Intelligent Tutoring Systems*, . New York: Springer-Verlag.

Hess, R. A. (1976). *A Method for Generating Numerical Pilot Opinion Ratings Using the Pilot Model* : NASA

Hess, R. A. (1977a). "Analytical Display Design for Flight Tasks Conducted under Instrument Meteorological Conditions". *IEEE Transactions on SMC*, SMC-7, 453-462.

Johnson, W. E., & Soloway, E. (1985). *Intention-Based Diagnosis of Programming Errors* (19): Yale University, Department of Computer Science, New Haven, CT

Jones, D. R., Abbott, T. S., & Burley, J. R. (1992). "Evolution of Conformal and Body-Axis Attitude Information for Spatial Awareness". *Helmet-Mounted Displays III, Proceedings of the International Society for Optical Engineering*, Orlando, FL.

Kaplan, R., & Rock, D. (1995). "New Directions for Intelligent Tutoring". *AI Expert*, February.

Kaplan, R., Trenholm, D., Gitomer, D., & Steinberg, L. (1993). "A Generalizable Architecture for Building Intelligent Tutoring Systems". *Proc. of Applications of Artificial Intelligence Conference: Knoweldge-Based Systems in Aerospace and Industry*, Orlando, FL.

Kimball, R. (1982). "A Self-improving Tutor for Symbolic Integration". In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, . London: Academic Press.

Kleinman, D. L., & Baron, S. (1971). *Analytic Evaluation of Display Requirements for Approach to Landing* (CR-1952): NASA (November).

Kleinman, D. L., Baron, S., & Levison, W. H. (1970). "An Optimal Control Model of Human Response, Part 1: Theory and Validation". *Automatica*, 6.

Kleinman, D. L., & Killingsworth, W. R. (1974). *A Predictive Pilot Model for STOL Aircraft Landing* (CR-2374): NASA (March).

Krishnakumar, K. S., Sawal, D., Bailey, J. E., & Dohme, J. A. (1991). "A Simulator-Based Automated Helicopter Hover Trainer-Synthesis and Verification". *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5), 961-970.

Levison, W. H., Baron, S., & Kleinman, D. L. (1969). "A Model for Human Controller Remnant". *IEEE Trans. on Man-Machine Systems*, 10.

Orey, M., & Nelson, W. (1993). "Development Prinicples for Intelligent Tutoring Systems: Integrating Cognitive Theory into the Development of Computer-Based Instruction". *Educational Technology Research and Development*, 41(1), 59-72.

Regian, J. W. (1989). "An Intelligent Instrument Flight Trainer". *Proceedings of AIAA Computers in Aerospace VII*, Monterey, CA.

Roscoe, S. N., & Williges, B. H. (1980). "Measurement of Transfer of Training". In S. N. Roscoe (Ed.), *Aviation Psychology*, . Ames, Iowa: Iowa State University Press.

Stengel, R. F. (1986). *Stochastic Optimal Control: Theory and Application*. (Vol. Wiley-Interscience): New York.

Talbot, P. D., & Corliss, L. D. (1977). *A Mathematical Force and Moment Model of a UH-IH Helicopter for Flight Dynamics Simulations* (NASA Technical Memo TM-73254)

Talbot, P. D., Tinling, B. E., Decker, W. A., & Chen, R. T. N. (1982). *A Mathematical Model of a Single Main Rotor Helicopter for Piloted Simulation* (NASA TM-84281) (September).

U.S. Army (1994). *Flight Training Guide: Th-67 Initial Entry Rotary Wing Aviator Course, Primary Phase*. U.S. Army Aviation Center, Fort Rucker, AL. (March).

Woolf, B., Blegen, D., Jansen, J., & Verloop, A. (1986). "Teaching a Complex Industrial Process". *Fifth National Conference on Artificial Intelligence*, Philadelphia, PA: AAAI.

Zacharias, G. L. (1985). *Modeling the Pilot's Use of Flight Simulator Visual Cues in a Terrain-Following Task* (R8505): Charles River Analytics Inc.

Zacharias, G. L., Barros, M. A., & Njie, F. D. (1994). *Hybrid Simulator-Based Intelligent Tutoring System for Medical Specialist Training* (Final Report R93072): Charles River Analytics (August).

Zacharias, G. L., & Brun, H. M. (1986). *Model-Based Methodology for Terrain-Following Display Design* (R8603): Charles River Analytics Inc. (February).

Zacharias, G. L., & Brun, H. M. (1987). *OCM-Based Descriptive Learning Model* (Technical Report R8703): Charles River Analytics Inc. (August).

Zacharias, G.L., Riley, E.W., Gonsalves, P.G., and Dohme, J.A. (1993). "Intelligent Flight Trainer for Initial Rotary Wing Training," SAE Paper 932536, *Presented at Aerotech '93*, Costa Mesa, CA.

Appendix A: Operation of IFT Software

This Appendix describes how to operate the software installed at the IFT facility at Fort Rucker, AL.

1) The following machines are used during system operation:

   disphost: The Iris hosting the Tigers instrument panel.

   ari486: The Micron 486 hosting the SIM.

   ari586: The Pentium 586 hosting the IFT.

   Talker: The Gateway 486 hosting the speech board.

   gateway2: The second Gateway 486 that hosts the LabView CIP code.

2) All five machines (SIM, CIP, IFT, Iris, and "Talker") should be up and running. The Image Generators (IGs) should be reset and ready to go (i.e., the status lights should read '99' and the hard drives should no longer be spinning).

3) On the Talker machine, type the following commands (shown in italics):

   c:\> *cd speech*

   c:\speech> *run1*

   This activates the text-to-speech routines and starts a program called **receive1.exe**, which receives messages from the IFT on the COM1 serial port and sends them to the speech board.

4) On the Iris, perform the following:

   4a) Log in as **tom**.

   4b) Type in the following commands at the prompt, in order:

   cd tigers/th67cdb

   su

   source environment

   LoadCdb

   InitAll

   exit

   startdsp

5) Launch the LabView CIP code on the **gateway2** machine. The sequence of operations here is:

5a) Launch Windows by typing *win* at the **c:>** prompt.

5b) Once in Windows environment, double click on LabVIEW icon in the LabVIEW workgroup.

5c) Click on File on the pull-down menu and choose the Open command.

5d). Find **CIP(w/iris).vi** by following this directory and path:

c:\LabVIEW\ift\cip\maincip.llb\CIP(w/iris).vi

Note: The default directory of LabVIEW software is set to **c:\LabVIEW\ift\cip\maincip.llb**, so this should be the current directory when the Open File command is evoked.

5e) To launch CIP top level code and display its front panel, double click on **CIP(w/iris).vi** .

At this point, the CIP is ready to go. The CIP program is activated after the SIM program (crasim.exe) starts, as described next.

6) The current directory in the SIM (Micron 486) machine should be **\cra_ift\sim\run**. Start the SIM by typing one of the following at the 486's prompt:

c:\cra_ift\sim\run> *crasim /c /i /p /v2* (2-channel IG)

or

c:\cra_ift\sim\run> *crasim /c /i /p /v1* (1-channel IG)

The SIM will then expect the CIP to connect. The following messages will appear on the screen:

c:\cra_ift\sim\run> *crasim /c /i /p /v2*

Waiting for CIP to connect

Ensure that the instrument panel is displayed on the Iris before starting the CIP. To run the CIP, click on the arrow on the top left of the screen, or press ^r ("control r") from the keyboard. Once TCP connections with SIM platform and Display Host are established, the Instrument Panel will auto launch on the Display Host screen. The user can resize the Instrument Panel to fit the whole screen. Note that before CIP main code is launched, both SIM and Display Host programs should be running. The the communication structure is set

such that CIP *initiates* the connection request, whereas the other two platforms *receive* the request. CIP opens the connection, and the other two are listening.

7) Once the CIP has connected, the SIM will expect the IFT to connect. Wait for the SIM's screen to indicate that it is expecting the IFT to connect. The following messages will appear on the SIM's screen:

c:\cra_ift\sim\run> *crasim /c /i /p /v2*

Waiting for CIP to connect

CIP has connected

Waiting 30 seconds for IFT to connect

The current directory on the IFT (Pentium 586) should be **\cra_ift\ift_p2\run**. There are several options that may be selected at runtime, as follows. Each option is separated by a single space. Type the following at the IFT's prompt:

c:\cra_ift\ift_p2\run> *iftreal [options]*

Available options:

/t Use trim routines (MUST BE SELECTED)

/o Use OCM helper (MUST BE SELECTED)

/r Enable auto-reset

/d Enable data recording

/m2 Hover mode (optional if performing hover)

/m3 Hover turn

/m4 Hover taxi

/m5a Departure maneuver (i.e., first segment of the traffic pattern)

/m5b Complete traffic pattern

/m5c Approach segment only of traffic pattern

/to Begin maneuver with takeoff

Options /m2, /m3, /m4, /m5a, /m5b, and /m5c are mutually exclusive. The individual commands to start the IFT in each of the available maneuvers are summarized as follows:

7a) *Hover*

\cra_ift\ift_p2\run> iftreal /t /o (Hover. The /m2 is optional in a hover)

\cra_ift\ift_p2\run> iftreal /t /o /d (Hover with data recording)

\cra_ift\ift_p2\run> iftreal /t /o /r (Hover with auto-reset)

\cra_ift\ift_p2\run> iftreal /t /o /r /d (Hover with auto-reset AND data recording)

7b) *Hover Taxi*

\cra_ift\ift_p2\run> iftreal /t /o /m4 (Basic Hover Taxi)

\cra_ift\ift_p2\run> iftreal /t /o /m4 /d (Hover Taxi with data recording)

\cra_ift\ift_p2\run> iftreal /t /o /m4 /r (Hover Taxi with auto-reset)

\cra_ift\ift_p2\run> iftreal /t /o /m4 /r /d (Hover Taxi with auto-reset AND data recording)

7c) *Hover Turn*

\cra_ift\ift_p2\run> iftreal /t /o /m3 (Basic Hover Turn)

\cra_ift\ift_p2\run> iftreal /t /o /m3 /d (Hover Turn with data recording)

\cra_ift\ift_p2\run> iftreal /t /o /m3 /r (Hover Turn with auto-reset)

\cra_ift\ift_p2\run> iftreal /t /o /m3 /r /d (Hover Turn with auto-reset AND data recording)

7d) *Departure Maneuver*

\cra_ift\ift_p2\run> iftreal /t /o /m5a (Basic Departure)

\cra_ift\ift_p2\run> iftreal /t /o /m5a /d (Departure with data recording)

\cra_ift\ift_p2\run> iftreal /t /o /m5a /r (Departure with auto-reset)

\cra_ift\ift_p2\run> iftreal /t /o /m5a /r /d (Departure with auto-reset AND data recording)

7d) *Traffic Pattern*

\cra_ift\ift_p2\run> iftreal /t /o /m5b (Basic Traffic Pattern)

\cra_ift\ift_p2\run> iftreal /t /o /m5b /d (Traffic Pattern with data recording)

\cra_ift\ift_p2\run> iftreal /t /o /m5b /r (Traffic Pattern with auto-reset)

\cra_ift\ift_p2\run> iftreal /t /o /m5b /r /d (Traffic Pattern with auto-reset AND data recording)

7d) *Final Approach*

\cra_ift\ift_p2\run> iftreal /t /o /m5c (Basic Approach)

\cra_ift\ift_p2\run> iftreal /t /o /m5c /d (Approach with data recording)

\cra_ift\ift_p2\run> iftreal /t /o /m5c /r (Approach with auto-reset)

\cra_ift\ift_p2\run> iftreal /t /o /m5c /r /d (Approach with auto-reset AND data recording)

8) Once the out-the-window visual display comes up and the IG's hard drive stops spinning, the system is ready to go. At this point, several startup initialization messages will appear on the SIM's screen, followed by "**Waiting for START signal...**".

9) DO NOT PRESS START YET!!! You must first select a flight maneuver from the menu options provided on the IFT display. The default choice (7) selects the maneuver last used. On the first startup, this will be the maneuver selected using the runtime options. This menu enables you to select a different maneuver after pressing RESET during flight. Once you have answered the questions posed the IFT, the IFT's screen will indicate that it is waiting for the START signal from the SIM.

10) Press the START button in the simulator cab to bring up the trim display.

11) Preposition the controls in accordance with the graphical display. Once the controls are positioned properly, the simulator will start (after a short delay).

12) Fly the simulator! Press the RESET button freeze the SIM. When you're ready to proceed, press START again to bring back the trim display.

13) To exit back to the system directly during operation, you may press ESC on either the 586 IFT or 486 SIM machine, or press the mouse button on the 486 CIP with the mouse cursor over the icon marked 'STOP'.

Appendix B: Software Overview

The main software elements of the IFT system are the cockpit interface processor (CIP), the simulator model (SIM), and the intelligent flight trainer itself (IFT). Each of these is now described.

B.1. Cockpit Interface Processor (CIP) Platform

This section describes the CIP platform characteristics, instructions to launch the platform, the sequence of operations in the CIP code, and the data packet structures that are sent and received between CIP and other nodes in the integrated system. Appendix C describes the wiring for switches and controls in the cockpit, and how they are connected to the CIP-based A/D board. Appendix D is the user guide for the Calibration Routine, which is an on-line routine for scaling sampled controls from the cockpit, and Appendix E describes the interface between CIP and Display Host (i.e., the SGI Iris), along with sequence of operations of the latter platform.

The CIP platform is directly interfaced with the OH-58 cab and communicates via ethernet with the SIM and Display Host platforms. One of the CIP's basic tasks is acquiring and sampling signals from the mechanical flight controls in the cockpit. Sampled data are then scaled and sent to the SIM via ethernet communication. The CIP also relays flight parameters received from the SIM to the Display Host platform (SGI 4D50), to be shown on the cockpit Instrument Panel. The CIP implements the trim logic, allowing the user to pre-position cockpit controls for flight during the trim routine. The CIP platform also provides the integrated system with a timer that synchronizes operations between nodes at 30 Hz per iteration.

The CIP hardware includes a 33 MHz 486 PC (Gateway 2000), 21" color monitor (Nanao FlexScan F760i.W), multi-function A/D board (NI, Lab-PC+), and a Network Adapter (Etherlink III, 3C509-Combo).

All CIP software was developed and implemented in LabVIEW development environment. The networking package used for the Window-based code was TCPOPEN, which has a dynamic link library most compatible with LabVIEW. The CIP code is structured modularly and hierarchically, and it is organized in compressed libraries that include data acquisition, networking and scaling libraries.

B.1.1 Software Modules & VI Libraries

The CIP directory under **c:\LabVIEW\ift** on the gateway2 machine contains the following VI (virtual instrument) libraries:

CALBRTN.LLB: Includes all VIs needed for the calibration of sampled control voltages. "Calibration Main.vi" is the top level VI for the calibration routine.

CIPTEST: Analog and digital test VIs, and "CIP Loop.vi", which is a stand alone CIP code.

DAQ.LLB: VIs for configuring the A/D board, and acquiring data from cockpit controls.

IRIS.LLB: VIs to interface and communicate with the Display Host.

MAINCIP.LLB: Top level VI (CIP(W/IRIS).vi), and top level VIs for the three modes of operation, and initialization and termination VIs.

NETWORK.LLB: VIs for TCP/IP network communication.

SCALING.LLB: VIs to scale the sampled control voltages.

TRIM.LLB: VIs for the Trim routine, excluding the top level VI.

UTILITY.LLB: VIs for Synchronizing and controling loop time.

B.1.2 Sequence of Operations & Data Packet Structures

Figure B.1 shows the sequence of operations for the CIP node. This platform communicates directly only with the SIM and Display Host; hence the read/write operations within the sequences in each mode only refer to these two other nodes. In addition, since the only task of Display Host within the integrated system is to display data graphically in the Trim and Flight mode, it only receives data packets from the CIP only.

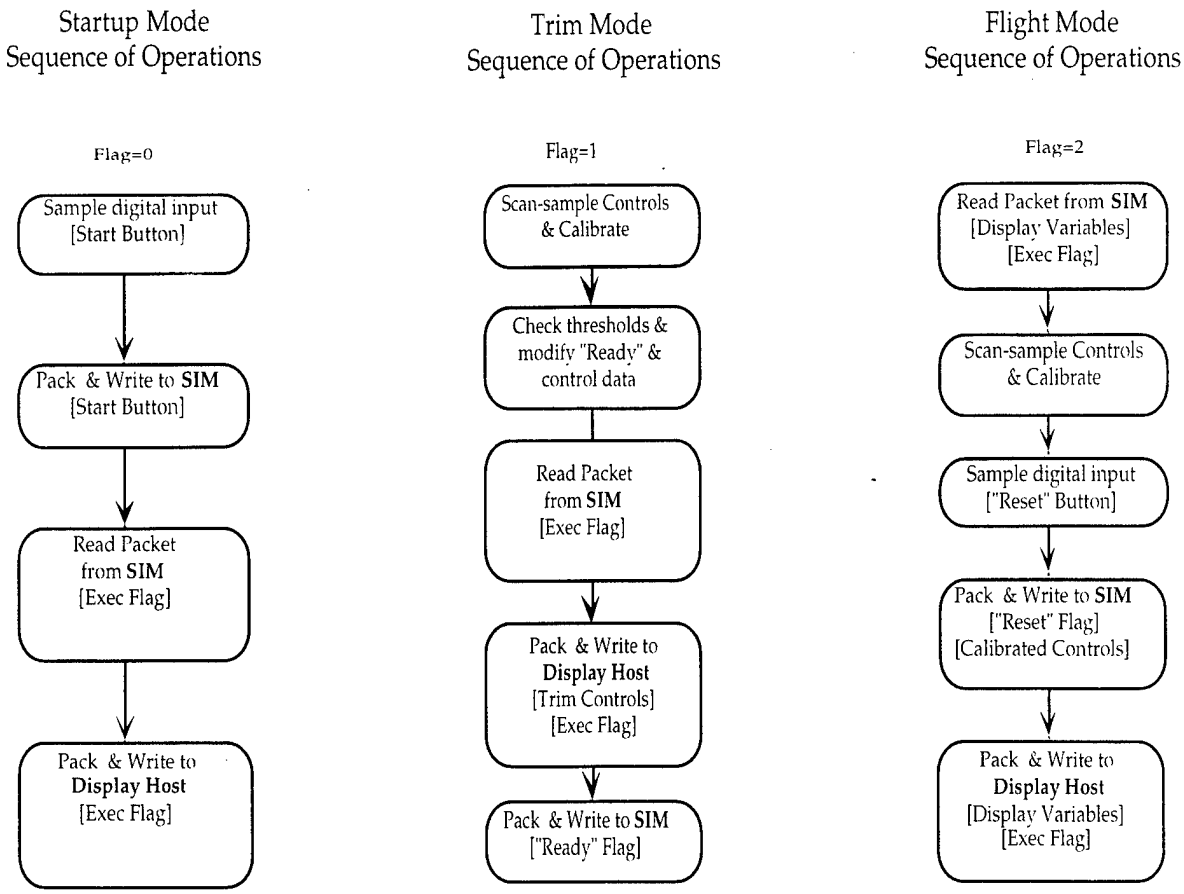
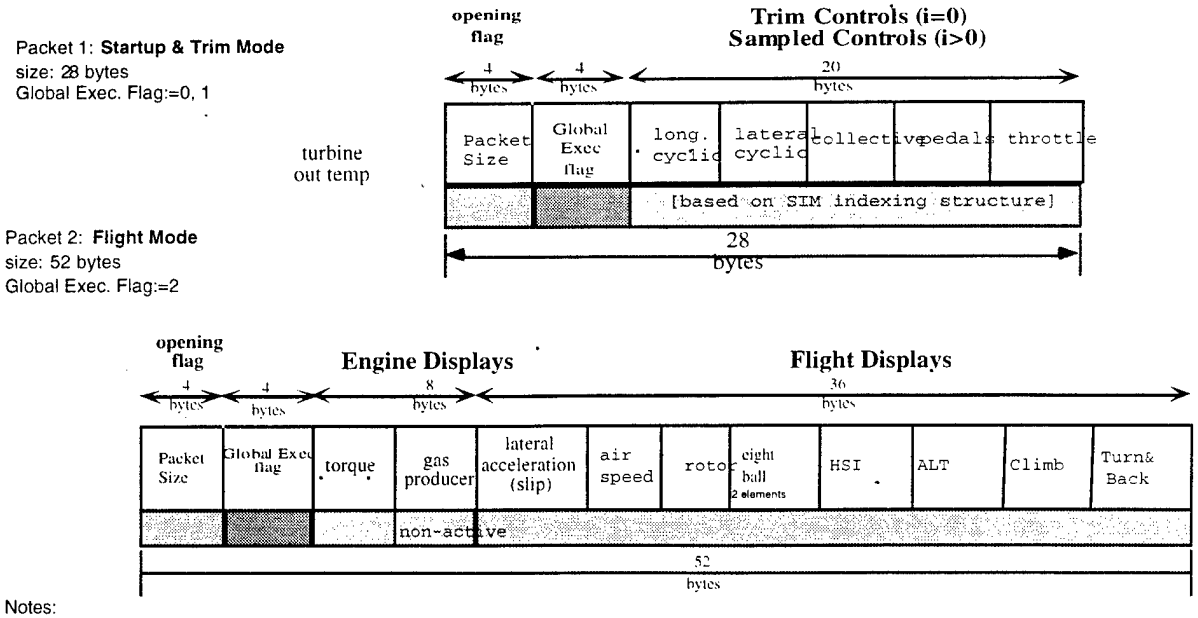


Figure B.1: Sequence of Operations on the CIP Platform

At each mode of operation, different packet structures with relevent data are sent from CIP to SIM and Display Host platforms. Figure B.2 shows the packets sent to Dispay Host during each mode of operation.

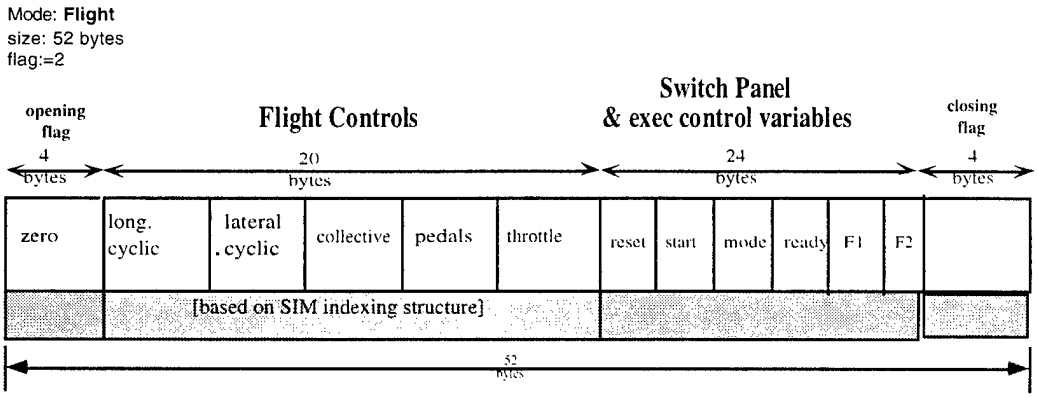


Notes:

(i) "Trim Controls" of Packet 1 is read once at the initialization stage of the Startup Loop.
(ii) When all controls are pre-positioned correctly, CIP sends "Global Exec Flag" = 2.
(iii) "Global Exec Flag" is read at each tick time (30 Hz) from the packet sent to control the executive logic of Display Host.
(iv) When "Global Exec Flag" = 2, Display Host will jump out of the Trim Routine, wait for three seconds, and launch the Instrument Panel.

Figure B.2: Data Packet Stuctures sent from CIP to Display Host

Figure B.3 illustrates the data packet structure sent from CIP to SIM at different modes.



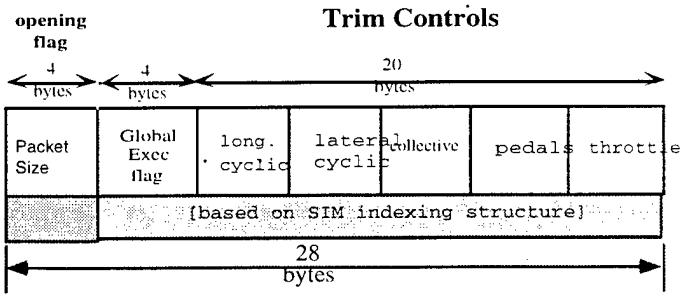
Notes:

(i) Flight Controls order is based on sim.h type definition
(ii) "reset", "start" and "mode" variables currently take digital inputs from switch panel in the cockpit.
(iii) F1 & F2 are currently undefined and do not perform any executive control tasks.

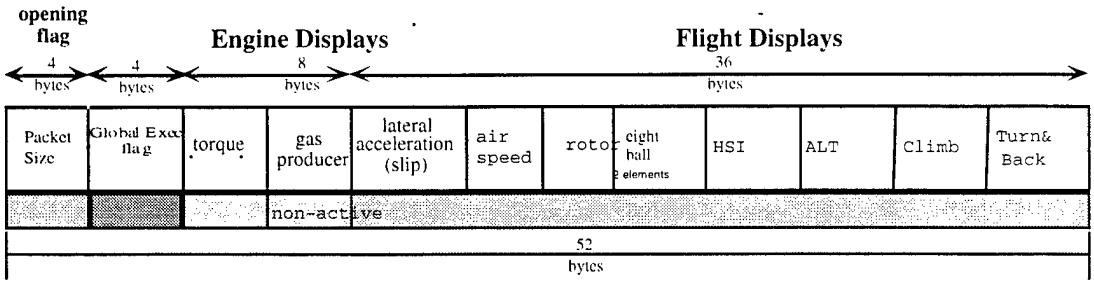
Figure B.3: Data Packet Stucture sent from CIP to SIM

Note that the data structure sent to Display Host at the Startup and Trim modes is identical to the data packet sent from SIM to CIP during these modes. Figure B.4 shows the data content and the packet structure sent from SIM to CIP at different modes of operation.

Packet 1: **Startup & Trim Modes**
size: 28 bytes
flag:=0, 1



Packet 2: **Flight Mode**
size: 52 bytes
flag:=2



Notes:

(i) Packet 1 is read once at the initialization stage of the Start-Up Loop.
(ii) If 4D50 platform hosts the Trim Routine, Packet 1 will also be sent to it.

Figure B.4: Data Packet Stucture sent from SIM to CIP

For the CIP/cockpit wiring, instructions to use the calibration routine, and the CIP/Display Host architecture, please refer to Appendices C, D, and E.

B.2 UH-1 Aerodynamic Model Simulator Host (SIM)

The source code for the vehicle model is on the Pentium (ari586), in the directory **c:\cra_ift\sim\source**. The simulator actually runs on the Micron 486 (ari486). The runtime files are located on that machine in the directory **c:\cra_ift\sim\run**. The executable file is called **crasim.exe**. A duplicate of the files in the runtime directory may be found on the Pentium, in a directory with the same name. Table B.1 lists the files that are needed to compile crasim.exe.

Table B.1: Source code files for the simulator program crasim.exe.

File	Description
crasim.mak	Visual C "makefile"
bbn_comm.c	Code to communicate with the BBN IGs. Has been modified to permit 1- or 2-channel operation at runtime.
bbn_main.c	Interface code between bbn_comm.c and main simulator shell.
ground_c.c	Ground contact model.
matrix.c	Matrix transformation routines needed by bbn_comm.c.
sendrecv.c	Ethernet communication routines.
sim.c	**Main simulator executive code.**
uasim.c	Helicopter flight model.
bbn.h, bbn_comm.h, bbn_defs.h, data_typ.h, ikon.h	Header files for the BBN interface code.
common.h	Definition of common constants used throughout the code.
ground_c.h	Header file for ground_c.c
sim.h	Header file for sim.c. Contains definition of data structures that are common between the SIM and the IFT.
uasim.h	Header file for uasim.c
likondrv.lib	Compiled libraries needed during linking by bbn_comm.c

Table B.2 lists the files located in the **c:\cra_ift\sim\run** directory (on the Micron 486). A duplicate of each of these files may be found on the Pentium in the same directory.

Table B.2: Runtime files for the simulator executable crasim.exe.

File	Description
crasim.exe	Executable Program
dbase.dat	Aerodynamic data file
ig.dat	Parameters for the image generator code. Contains the parameters describing the rotation from the front to the side window in the helicopter cab.
initial.dat	Default initial conditions used to initially position the helicopter over the runway at startup.

The helicopter's default initial conditions are stored in the file "initial.dat", located in the runtime directory. This file contains the following (floating-point) parameters, in order:

Gust level (set to zero)
Wind velocity (set to zero)
Wind direction (set to zero)
x position, ft
y position, ft
z position, ft (negative above sea level)
Initial forward velocity, ft/sec

Heading, radians (should be zero to line up with runway)

B.2.1 Compiling the SIM Source Code

The source code is readily compiled from within the Visual C environment under Microsoft Windows on the Pentium (ari586). First, start Windows by typing "win" at the DOS prompt. Under the "Main" window, you should see an icon for Visual C. Double click on it with the mouse to bring up the environment. Under the **Project** menu, select **Open...**. This will bring up a *browser*, which you can use to navigate to the **\cra_ift\sim\source** directory (with the mouse). In the browser's left window, a file called **crasim.mak** will appear. This is the so-called "makefile" for the simulator. Double-click on it. You can now edit the source code associated with the project by going to the **File** menu. In order to compile the executable after making changes to the source code, select **Build CRASIM.EXE** from the **Project** menu. All compiler and linker flags have been preset to the appropriate values. Assuming that there are no compiler errors (often a specious assumption) and you want to run the sim, you must first exit Visual C and Windows. To exit Visual C, select **Exit** from the **File** menu. Then exit Windows and return to DOS.

The resultant executable crasim.exe will be placed in the **c:\cra_ift\sim\source** directory on the Pentium. Prior to execution, it must be moved to the **c:\cra_ift\sim\run** directory on the Micron 486. The new executable may be transferred from the Pentium to the 486 using ftp (file transfer protocol). The procedure is as follows:

1. If in Windows on the Pentium, exit to DOS.
2. Change the current directory to **\cra_ift\sim\source** on the Pentium.
3. Type *ftpd* at the Micron 486 (SIM machine) prompt.
4. If the SIM has been recompiled, there will be a new **crasim.exe** file in the **\cra_ift\sim\source** directory on the Pentium. Whether or not the crasim.exe file is new may be verified by inspection of the file creation date (using the "dir" command).
5. Make an ftp connection to the 486 by typing the following at the Pentium's prompt:

   c:\cra_ift\ift_p2\source> *ftp ari486*

6. The ftp program will ask you for a username. Type "sim" (without quotes!) in response. It will then ask you for a password. Again, type "sim". You will automatically be connected to the proper directory on the 486 (**c:\cra_ift\sim\run**). Change the file type to binary by typing "binary". Then, type "put crasim.exe" to transfer the file. Finally, type "quit" to exit ftp. On the 486, hit "D" (lowercase OK) to exit ftpd. The new crasim.exe executable file is now situated in the correct directory on the 486 and is ready to run. See Appendix A for instructions on bringing up the entire system.

Alternatively, the user may use a floppy disk to transfer the file crasim.exe to its correct location on the Micron 486.

B.3. Intelligent Flight Trainer (IFT) Software

Like the simulator model, the IFT code was developed under Microsoft Visual C. The IFT contains two main computational elements -- the OCM-based *helper*, which augments pilot inputs as a function of average hover proficiency, and the CLIPS expert system that provides voice feedback functions. At the time of writing, the expert system contains modules for training hover, hover taxi, hover turn, traffic pattern (and subsets thereof), and land from hover.

Runtime files are located in the directory **c:\cra_ift\ift_p2\run** on the Pentium (ari586). The IFT code has recently been modified to make use of the PharLap Dos Extender, to make more memory available to the application (normal Dos programs have a memory limit of 640K). This has entailed dividing up the application into two executable files: **iftreal.exe**, and **iftprot.exe**. The program iftreal.exe is invoked by the user (as described in Appendix A), and it starts iftprot.exe. This modification was necessary because the Ethernet and serial port libraries used by the IFT do not work from protected mode (i.e., extended memory). Therefore, all subroutines that must reside in so-called real mode are placed in the real-mode program **iftreal.exe**. This program calls the protected mode (extended memory) program **iftprot.exe**, which establishes pointers to the real-mode functions using techniques described in the Dos Extender manuals. All of the IFT's main computational elements are contained within the program iftprot.exe; any changes made to the software will most likely be in this program. Table B.3 lists all of the files that must be located in the **c:\cra_ift\ift_p2\run** on the Pentium at runtime.

Table B.3: IFT Runtime files.

File	Description
CLIPS rulebases:	
hover.clp	Hover-specific rulebase.
htaxi.clp	Hovertaxi-specific rulebase.
hturn.clp	Hoverturn-specific rulebase.
main.clp	CLIPS rulebase file used in all flight modes. Contains landing rules and thresholds that define an acceptable landing.
traffic.clp	Rulebase for the traffic pattern.
IFT parameter files:	
ift_mesg.dat	IFT message database. The first line is a 0 or 1, depending on whether or not the speech board is to be used. The value of the flag is ignored. The second line is the minimum delay in seconds between consecutive messages. This value is also ignored.
ift_parm.dat	IFT runtime parameters. Each one has a description next to it. This file is discussed in greater detail below.
trim.hov	Hover trim state and control values.
Helper files:	
ocm.hov	OCM gains for the hover mode.
ocm.tax	OCM gains for the hover taxi mode.
ocm.to	OCM gains for takeoff (not used).
ocm.tun	OCM gains for the hover turn mode.
latgain.ocm	Lateral gains for the new "flight" helper
longain.ocm	Longitudinal gains for the new "flight" helper
trimgrid.dat	Two dimensional trim data specifying trim conditions as a function of altlitude and airspeed. Used by the gain-scheduled "flight" helper.
Other files:	
dbase.dat	Aerodynamic data base file. Used for trim computations. Identical to the dbase.dat file used by the SIM.
reset.hov	Auto-reset parameters for the hover maneuver.
reset.tax	Auto-reset parameters for the hover taxi maneuver.
reset.tp	Auto-reset parameters for the traffic pattern.
reset.tun	Auto-reset parameters for the hover turn.

The file that will be changed most often on the IFT is ift_parm.dat. As such, it is now considered in greater detail. Typical contents of this file are as follows. There are a total of 19 parameters that may be set by the user.

```
1)    150 WINDOW_SIZE (based on 30 Hz update rate!)
2)    0 MANEUVER (0 = hover, 1 = taxi, 2 = climb, 3 = turn)
3)    2.0 JPLON_UPPER was 1.5
```

```
4)    1.0 JPLON_LOWER was 0.75
5)    1.2 JPLAT_UPPER was 1
6)    0.6 JPLAT_LOWER was 0.5
7)    6 Initial NPLON (new OCM)
8)    6 Initial NPLAT (new OCM)
9)    10 NP_UPPER maximum allowable NP value (new OCM)
10)   1 NP_LOWER minimum allowable NP value (new OCM)
11)   20 time(sec) between automatic level changes
12)   131688.0 normal hover X position (ft)
13)   148879.0 normal hover Y position (ft)
14)   -761.1 normal hover Z position (negative height in ft)
15)   0.0 normal hover PSI angle (angle of the runway in degrees)
16)   8.0 Desired hover taxi speed (ft/sec)
17)   20 Initial Delay for IFT (sec) during which no scoring takes place
18)   20 time(sec) required at minimum help level to pass
19)   -756.1 z position of the ground (negative height in ft MSL)
```

These parameters are defined as follows:

1) WINDOW_SIZE is the size of the data window used to average the data sent to the CLIPS rulebase. At a 30 Hz update rate, a window size of 150 implies that the data sent to the IFT has been averaged over 5 seconds. If the advisor "lags" in its response; i.e., the advice is a bit behind the student's actual position, try reducing the window size.

2) MANEUVER is the flight maneuver. This is no longer used.

3) JPLON_UPPER is the upper limit on the longitudinal OCM cost function. If the student's performance is such that JPLON stays above this value for a certain length of time (see item 11), the help level increases.

4) JPLON_LOWER is the lower limit on the longitudinal OCM cost function. If the student's performance is such that JPLON value stays below this value for a certain length of time, the help level decreases.

5,6) JPLAT_UPPER and JPLAT_LOWER are the corresponding values for the lateral cost function JPLAT.

7,8) Initial NPLON and Initial NPLAT are the help levels at startup.

9) NP_UPPER is the maximum help level allowable.

10) NP_LOWER is the lowest help level allowable. Once the student meets the performance requirement at this help level, he/she "passes."

11) The time in sec is the interval between automatic help level changes (either increases or decreases). If either JPLON > JPLON_UPPER or JPLAT > JPLAT_UPPER for this duration, help level increases. If both JPLON < JPLON_LOWER and JPLAT < JPLAT_LOWER for this duration, help level decreases.

12-15) The next four numbers are the desired hover position coordinates within the BBN Image generator's database. The nominal z coordinate of -761.1 ft is 5 ft AGL at the south end of the runway.

16) The Desired hover taxi speed is the forward speed that the student is expected to match during the hover taxi maneuver.

17) The Initial Delay for IFT is a time-out at the beginning of a run during which no IFT scoring takes place. This is not yet used.

18) The last parameter is the time required at the minimum help level (which is set above) to pass the maneuver. Once the student maintains JPLON < JPLON_LOWER and JPLAT < JPLAT_LOWER for this duration at the minimum help level, the IFT tells the student that he/she has "passed" the maneuver.

19) z coordinate of ground position at the hover location. This is negative altitude in ft MSL.

B.3.1 Compiling the IFT Source Code

The source code for the IFT resides on the *Pentium* in the directory **\cra_ift\ift_p2\source**. Table B.4 lists the files needed to compile the executable file iftreal.exe, and Table B.5 lists the files needed for iftprot.exe.

Table B.4: Source code for iftreal.exe

File	Description
iftreal.mak	Visual C "makefile"
common.c	Numerous utility routines used by the main code.
iftreal.c	Main program, containing real-mode functions.
common.h	Header file for common.c
nfswrap.h	Headers for the real-mode functions
sim.h	Contains type definitions and data structure definitions used by the SIM and IFT code.

Table B.5: Source code files for iftprot.exe

File	Description
iftprot.mak	Visual C "makefile"
common.c	Utility print routines used by several different modules
data.c	Processes incoming helicopter state information and sets IFT modes
htaxi.c	Computational modules for the hover taxi maneuver
hturn.c	Computational modules for the hover turn maneuver
iftprot.c	**Main executive program**
ift_clip.c	Interface between CLIPS and C
message.c	Loads IFT message database and acts as intermediary between CLIPS and serial port communication routines
nfswrap.c	Sets up pointers to the real-mode code (in iftreal.exe) that must be accessed from iftprot.exe
nr2.c, nrutil.c	Various vector/matrix routines
ocm.c	OCM helper code (Old helper, for hover maneuvers)
ocm2.c	New helper code (gain scheduled helper), used in traffic pattern
printdat.c	Routines for printing various information to the screen (not used)
record.c	Routines for storing data to disk during a run
reset.c	Auto-reset logic modules
setup.c	Initial setup and IFT configuration routines
traffic.c	Comptutational modules for the traffic pattern maneuver
trim.c	Trim routines
trimdata.c	Trim routines
uasim.c	Helicopter model code, used by trim routines
window.c	Code for displaying the operator's console during operation
common.h	Header file for common.c
common2.h	Definition of common constants (pi, etc.)
data.h	Header file for data.c
htaxi.h	Header file for htaxi.c
hturn.h	Header file for hturn.c
ift.h	Header file for ift.c
iftsetup.h	Header file for setup.c
ift_clip.h	Header file for ift_clip.c
message.h	Header file for message.c
nfswrap.h	Header file for nfswrap.c
ocm.h, ocm2.h	Header files for ocm.c and ocm2.c, respectively
record.h	Header file for record.c
reset.h	Header file for reset.c
sim.h	Definition of data structures that are common between the SIM and the IFT
traffic.h	Header file for traffic.c
trimdata.h	Header file for trimdata.c
uasim.h	Header file for uasim.c
window.h	Header file for window.c

The procedure for opening the project works the same as that for the sim -- start Windows (on the Pentium), run Visual C, and select select **Open...** from the **Project** menu. Use the browser to navigate to the **c:\cra_ift\ift_p2\source** directory, and double click on **iftreal.mak** or **iftprot.mak** (the makefiles). If the code is recompiled, a new executable called **iftreal.exe** (or **iftprot.exe**) will be placed in the same directory. Once the program is compiled, exit back to DOS. The new executable files iftreal.exe or iftprot.exe will be in the directory **c:\cra_ift\ift_p2\source**. They must be copied to the directory **c:\cra_ift\ift_p2\run** before the system can be started. If the current directory is **c:\cra_ift_p2\source**, the following commands will accomplish the task:

c:\cra_ift_p2\source> *copy iftreal.exe ..\run*

copies **iftreal.exe** to the correct directory, or

c:\cra_ift_p2\source> *copy iftprot.exe ..\run*

copies **iftprot.exe** to the correct directory.

Once the new files have been placed in the run directory, the system may be started as described in Appendix A.

Appendix C: CIP/Cockpit Wiring Interface

Figure C.1 shows the schematic diagram for the wiring interface of CIP with the cockpit pushbutton switches. Figure C.2 shows the location of individual switches on the cockpit panel.

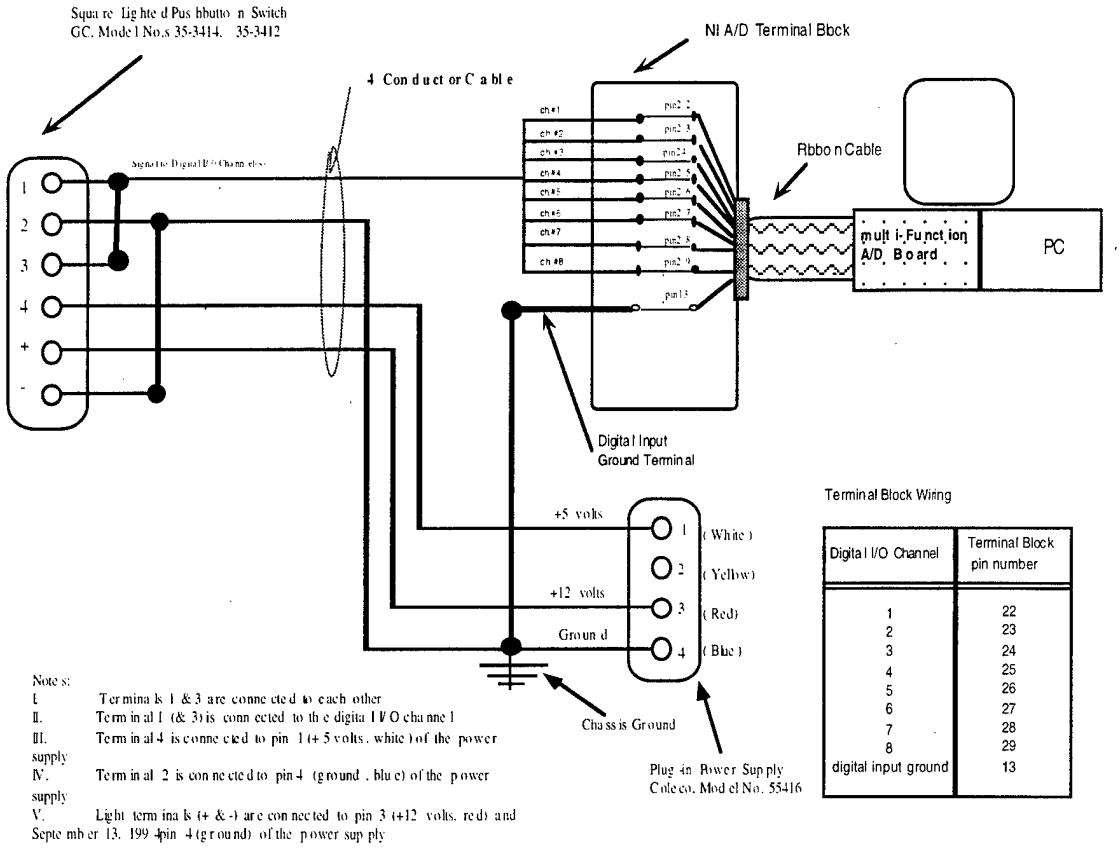
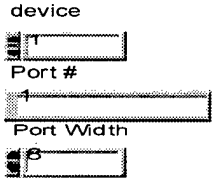


Figure C.1: Schematic diagram for the wiring interface of CIP with cockpit switches

Notes:

I. Slots 2, 3 & 4 contain no switches.
II. Pin 13 on Terminal Block is the Digital Ground for all the digital signals.
III. Switch 1 is latching, & 5 thru 8 are momentary.
IV. Digital port B (pins 22 thru 29) is configured for digital input sampling, corresponding to the following initialization parameters in LabVIEW:



V. For future reference, bit 8 of Digital port C is non-functioning. Also, Digital port A is configured for output, not input.

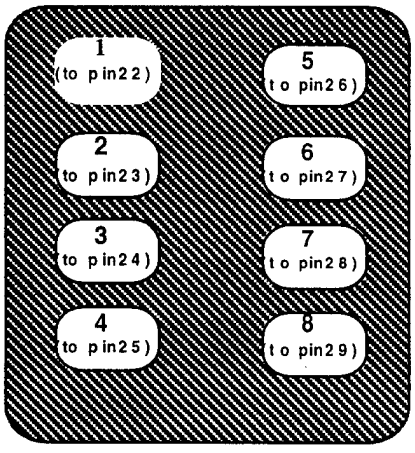


Figure C.2: Schematic Diagram of Cockpit Switch Panel

Appendix D: Calibration Routine User Guide for CIP Platform

The following is a set of instructions to use the Calibration Routine.

1. Launch Windows by typing win at the **c:>** prompt.
2. Once in Windows environment, double click on LabVIEW icon in the LabVIEW workgroup.
3. Click on File on the pull-down menu and choose the Open command.
4. Find Calibration **Main.vi** by following this directory and path:
   c:\LabVIEW\ift\cip\CALBRTN.llb\Calibration\Main.vi

   Note: The default directory of LabVIEW package is set to **c:\LabVIEW\ift\cip**, so this should be the current directory when the Open File command is invoked.
5. To launch Calibration Routine top level code and display its front panel, double click on **Calibration Main.vi**. The front panel will appear on the screen as shown in figure D.1.

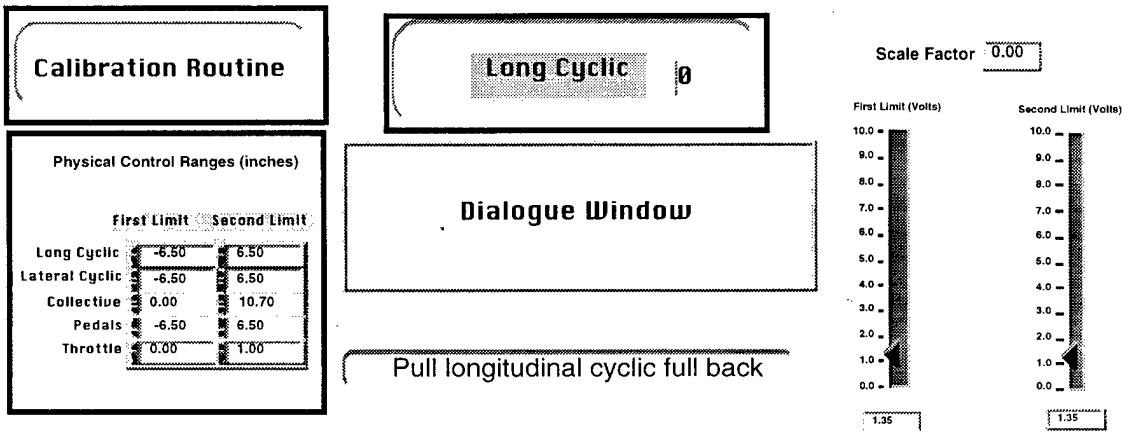


Figure D.1: Front panel of Calibration Main.vi

6. Before launching Calibration Main.vi, make sure that the **Physical Control Ranges**, displayed on the left side of screen, are the correct values and do not need modification. Once the routine is launched, the file containing the old values for these ranges **will be erased** and values shown on the front panel of the Calibration Main.vi will be stored in it. To make the changes on the front panel permanent, go to edit mode by pressing ^m (or "control m"), pull down **Operate** menu and choose "Make Current Values Default" command. Finally, save the VI by pulling the **File** menu and invoking the **save** command. This will ensure that when running Calibration Main.vi, the edited values will be used during the calibration and stored in the data file. The name of the data file containing the ranges is "Ranges.txt". The default range values are saved in Ranges.000.

Caution: These values will be used later in the real-time CIP main program. Make sure they are the correct values before running the calibration routine.

7. Launch Calibration Main.vi by selecting **Run** from under **Operate** menu. Interactive dialogue windows will appear on the screen instructing the user to set the position of each control twice, once for the first limit and then for the second limit. Below the dialogue window, each specified limit is defined by the type of physical action required by the user. For example, for the longitudinal cyclic, the user will be instructed to first "pull longitudinal cyclic full back", and then to "pull longitudinal cyclic full forward".
8. The user will press the **proceed** button on the dialogue window every time the control is set at the specified position. Once this button is pressed, the routine will take 100 voltage samples for averaging, and the vertical slide bar will show the corresponding voltage value during sampling. The voltage value is kept on the indicator until the next reading takes place.
9. The sequence of controls to be calibrated are long cyclic, lateral cyclic, collective, pedals, and throttle. The name of the control to be adjusted will be shown on top of the dialogue window as the user proceeds through the routine.
10. The indicator on the top right of the screen will show the value of the scale value currently calculated. The units are inches/volts. This value will be updated as the user moves to the next control. As the scale factor for each control is calculated, it will be written to a file one at a time. The name of the file is "scales.txt". This file will later be read by the real-time CIP main program during the initialization process.

    Note: Once the routine is launched, the file containing the old scale factors will be erased and values calculated by Calibration Main.vi will be stored in it. If you want to access the old values, they are saved in "scales.000".

    Caution: These values will be later used in the real-time CIP main program. Make sure they are the correct values before running the real-time CIP main program.

Appendix E: Ethernet Communication and Sequence of Operations on Display Host (SGI 4D/50)

E.1. Ethernet Communication

For proper communication between CIP and Display Host, please make sure that when configuring your Ethernet adapter, the Transceiver Type is set for on-board Coax (BNC), which is suited for thin Ethernet cable for **peer-to-peer connection**.

To make connection between the two platforms possible, when configuring the adapter, define the Ethernet address of the localhost and the CIP platform in the **hosts** file as follows:

```
# internet_addr     name          comments
192.9.200.98        localhost     DISPHOST #the Display Host IP address and name
192.9.200.99        Gateway2      #the The CIP IP address and name
```

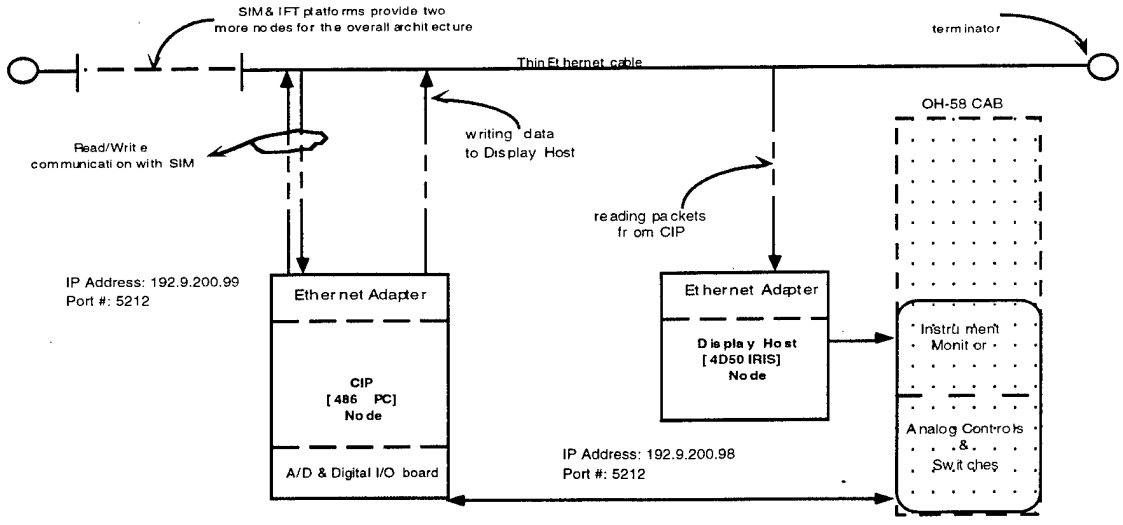
This information might be written to the host file during configuration of your Ethernet communication software, in which case hand-editing the file becomes unnecessary. The first line above defines 192.9.200.98 as the 4D50 platform's IP address and gives it a name as "DISPHOST". In the same manner, the second line in the hosts file imply that the CIP platform will open TCP connection with Display Host by searching for 192.9.200.98 address of a platform named DISPHOST.

The port # for the Ethernet communication between CIP and Display Host is assigned to 5214. When 4D50 Iris opens TCP connection with the CIP platform on 192.9.200.99, the port number for this connection should be 5214, otherwise no data transfer between nodes will occur. The Ethernet architecture between the two platforms is shown in Fig. E.1.

E.2. Sequence of Operations on 4D50 Iris

Figure E.2 shows the sequence of operations on the Display Host. As shown in the figure, the flag number read from the packet sent by CIP determines which set of operations the Display Host will be performing. The Global Executive Flag is sent within the packet at every update, which occurs at 30 Hz.

When the flag=0, no other data is read from the CIP, and no graphics is updated on the screen. This is the stand-by mode, where the main code on the Iris is running, but due to the state of the executive logic dictated by CIP to the Iris machine, the system is neither in the trim, nor in the run-time mode.



Notes:
(i) The overall structure is based on peer-to-peer architecture, allowing distributed processing between platforms.
(ii) The Transciever type for all ethernet adapters at each node should be configured for on-board Coax (BNC).

Figure E.1: Ethernet Architecture between CIP (486 PC) and Display Host (4D50 IRIS)

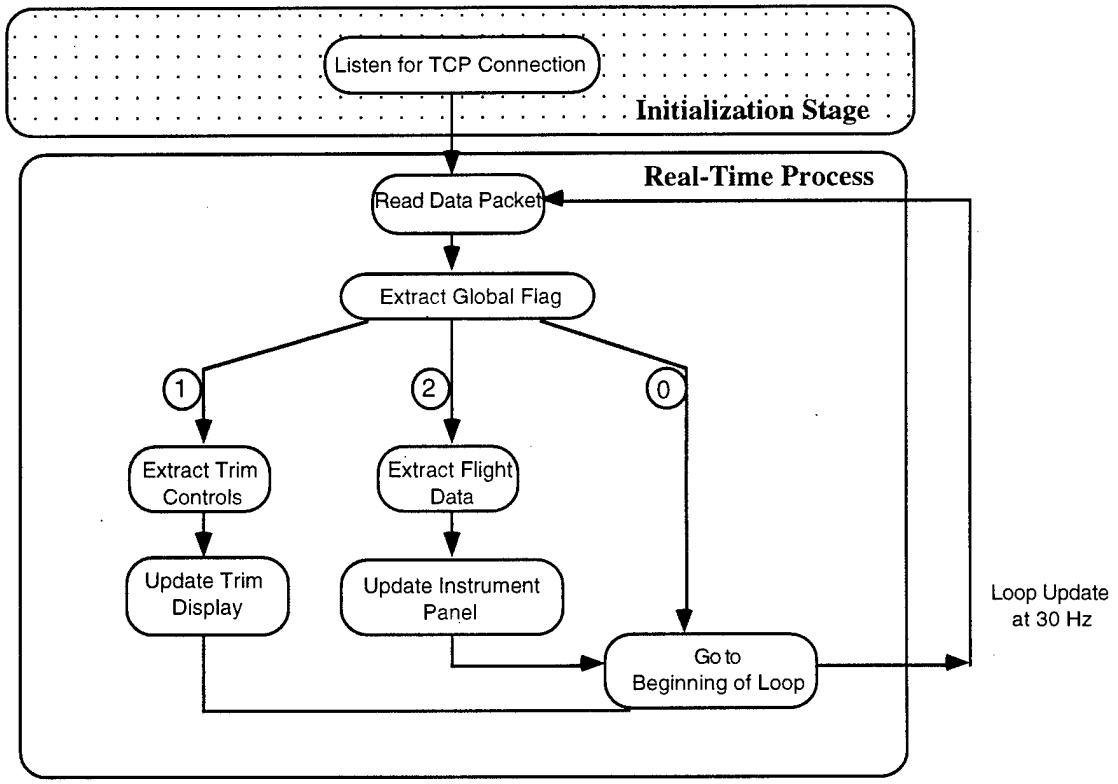


Figure E.2: Sequence of Operations of the Display Host [4D50 Iris]

When the flag=1, the Display Host will go into Trim mode, and at the first iteration, will read the desired trim values, and from then on, will read the sampled control data sent from CIP. The

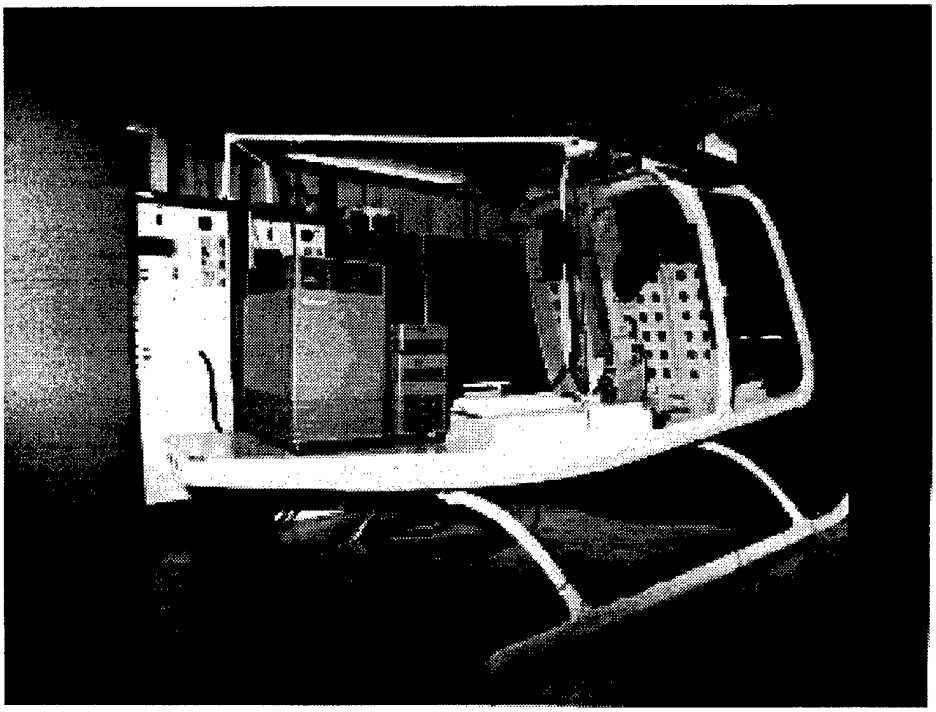


Figure F.2: OH-58 Cab with SGI Workstation and BBN Image Generators

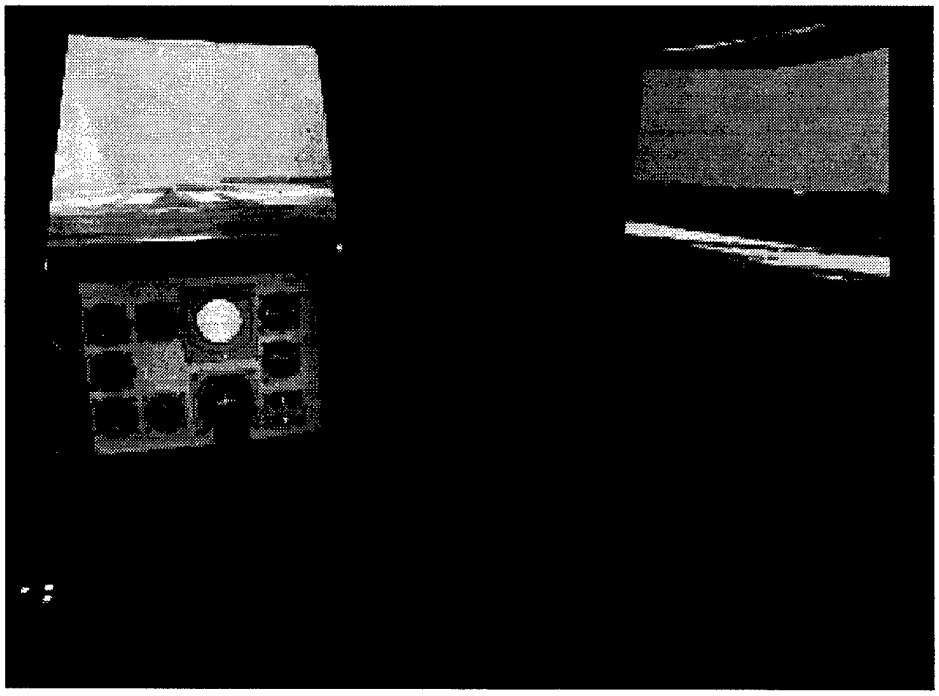


Figure F.3: Out-the-Window Display and Instrument Panel